**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК ___004.043_____

«До захисту допущено»

Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2019 р.

# Магістерська дисертація

**на здобуття ступеня магістра**

зі спеціальності 126 «Інформаційні системи та технології»

на тему: Система оптимального налаштування штучних нейронних мереж глибокої довіри

Виконав (-ла): студент (-ка) шостого курсу, групи ІК-з81мп
(шифр групи)

Марусик Олександр Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Чумаченко О. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент доцент каф. АКІК НАУ, к.т.н., доцент Пантєєв Р.Л. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
«___»_____2019 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Марусику Олександру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Система оптимального налаштування штучних нейронних мереж глибокої довіри»
_____,
науковий керівник дисертації доцент, к.т.н., доцент Чумаченко О.І.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від « 05 » 11 2019 р. № 3836-с

2. Термін подання студентом дисертації 18.11.2019 р.

3. Об'єкт дослідження Штучні нейронні мережі глибокого навчання
_____

4. Предмет дослідження Алгоритми структурно-параметричного синтезу нейронних мереж глибокого навчання
_____

5. Перелік завдань, які потрібно розробити аналіз предметної області і постановка задачі; огляд існуючих рішень; проведення порівняльних експериментів; розробка алгоритмів структурно-параметричного синтезу нейронних мереж глибокого навчання; розробка програмного продукту.

6. Орієнтовний перелік ілюстративного матеріалу блок-схема алгоритму структурно-параметричного синтезу нейронних глибоких мереж довіри; схема програмного засобу; результати експериментів.

7. Орієнтовний перелік публікацій _____
_____
_____

8. Консультанти розділів дисертації

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |

9. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 13.09.2019 р. | |
| 2 | Постановка задачі | 15.09.2019 р. | |
| 3 | Аналіз інформаційного забезпечення | 20.09.2019 р. | |
| 4 | Розробка алгоритмічного забезпечення | 15.10.2019 р. | |
| 5 | Розробка програмного забезпечення | 01.11.2019 р. | |
| 6 | Проведення експериментів | 05.11.2019 р. | |
| 7 | Маркетинговий аналіз стартап-проекту | 10.11.2019 р. | |
| 8 | Висновки | 15.11.2019 р. | |

Студент _____ О.М. Марусик
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ О.І. Чумаченко
(підпис) (ініціали, прізвище)

# АНОТАЦІЯ

Об’єктом розробки є система оптимального налаштування штучних нейронних глибоких мереж довіри.

Мета розробки – автоматизація структурно-параметричного синтезу нейронних мереж глибокого навчання, зокрема глибоких мереж довіри.

Робота присвячена розробці програмного засобу, що автоматизує структурно-параметричний синтез штучний нейронних мереж, та дозволяє отримати нейромережеві моделі, придатні для виробничого застосування. У дипломі розглянуті сучасні підходи до вирішення поставленого завдання. На їх основі розроблено архітектуру програмного застосунку, який являє собою програмну систему, яка складається з окремих компонентів, що робить її гнучкою для змін, легко керованою та дає можливість інтеграції з іншими системами. Застосунок побудований із використанням сучасних програмних засобів, може бути розгорнутий на широкому спектрі обчислювальних систем, є платформонезалежним.

Отримані результати можуть бути корисними при застосуванні у робототехнічних системах, що потребують розв’язання задачі розпізнавання зображень.

Ключові слова: нейронні мережі глибокого навчання, глибока мережа довіри, обмежена машина Больцмана, оптимізація глибокого навчання, автоматизована система, програмний застосунок.

Робота містить 89 аркушів, 15 рисунків, 45 таблиць, посилання на 32 джерел, 5 демонстраційних плакатів.

## SUMMARY

The object of development is a system of optimal setting of deep belief networks for using in robotic systems.

The goal of development is automation of structural-parametric synthesis of deep neural networks, in particular deep belief networks.

The work is devoted to the development of software that automates the structural-parametric synthesis of artificial deep neural networks, and allows to obtain neural network models suitable for industrial use. The diploma examines modern approaches to solving this task. Based on this, a software application architecture has been developed, which is a software system that consists of individual components, which makes it flexible for change, easily manageable and enables integration with other systems. The application is built using modern software tools and frameworks, can be deployed on a wide range of computing systems, is platform independent.

The results can be useful when applied to robotic systems that need to solve the problem of image recognition.



Keywords: deep neural networks, deep learning, deep belief networks, restricted Boltzmann machine, deep learning optimization, automated system, software application.


The work contains 89 pages, 15 pictures, 45 tables, links to 32 sources, 5 demo posters.

# Пояснювальна записка

## до магістерської дисертації

на тему: *Система оптимального налаштування штучних нейронних мереж глибокої довіри*

Київ – 2019 року

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ГМД – глибока мережа довіри.

ОМБ – обмежена машина Больцмана.

БШП – багатошаровий персептрон.

СОНГМД – система оптимального налаштування глибоких мереж довіри.

SGD – алгоритм стохастичного градієнтного спуску.

SGDM – алгоритм стохастичного градієнтного спуску з моментом.

NAG – алгоритм прискореного градієнту Нестерова.

AdaGrad – алгоритм адаптивного градієнту.

AdaM – алгоритм оцінки адаптивного моменту.

NAdaM – алгоритм адаптивного моменту Нестерова.

RMSProp – алгоритм розповсюдження кореня середнього квадрату градієнта.

PT – алгоритм паралельного відпуску.

CD – алгоритм contrastive divergence.

PCD – алгоритм persistent contrastive divergence.

MCMC – алгоритм Monte Carlo Markov Chain.

# ВСТУП

Нейромережеві технології сьогодні переживають бурхливий розвиток та застосовуються у найрізноманітніших областях для моделювання складних систем, які описуються мільйонами характеристик, що унеможливлює моделювання іншими методами, зокрема аналітично.

Це явище не оминуло й робототехніку, де штучні нейронні мережі застосовуються для широкого спектру задач: прийняття рішення, адаптивної поведінки у складному середовища, розпізнавання візуальних та аудіальних образів, фільтрації, стиснення, генерування сигналів тощо.

Побудова нейромережевих моделей досі залишається доволі складною задачею, зважаючи на специфіку предметної області, велику кількість параметрів моделей, та підходів щодо їх налагодження (навчання). Досі не знайдено універсального методу розв’язання цієї проблеми, що приводить до необхідності досліджувати кожну задачу окрему та створювати моделі вручну, а у кращому варіанті напівавтоматично.

Проте у всьому розмаїтті проблем та задач можна виділити групи схожих за призначенням та реалізацією задач, що придатні до повної автоматизації із отриманням моделей, без участі дослідника та придатних до подальшого практичного використання за призначенням у реальних системах.

Одним із прикладів таких задач є розпізнавання та класифікація візуальних образів для системи прийняття рішень у роботизованій агротехнічній системі, що займається автоматичним сортуванням врожаю.

У даній роботі розглядається система автоматичної побудови та налаштування глибокої нейромережевої моделі на основі штучної нейронної глибокої мережі довіри.

# РОЗДІЛ 1. НЕЙРОННА ГЛИБОКА МЕРЕЖА ДОВІРИ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ ГЛИБОКОГО НАВЧАННЯ

## 1.1. Нейронна глибока мережа довіри, структура та використання

Глибока мережа довіри (ГМД, deep belief network, DBN) – глибока генеративна нейронна мережа, що призначена для знаходження або відтворення розподілу ймовірності у даних, містить множину шарів латентних змінних, які мають зазвичай бінарну форму; вхідний шар може бути як бінарним, так і неперервним. Вони були винайдені Хінтоном у 2006 році після розробки ефективного алгоритму навчання обмеженої машини Больцмана [1][2].

ГМД із $n$ прихованими шарами містить $n$ матриць вагів $W^{(1)} \dots W^{(n)}$ та $b+1$ векторів зміщень $b^{(0)} \dots b^{(n)}$, де $b^{(0)}$ – зміщення видимого шару.

Математична модель ГМД має наступний вигляд [3]:

$$p\left(h^{(n)}, h^{(n-1)}\right) = \exp\left(b^{(n)\top} h^{(n)} + b^{(n-1)\top} h^{(n-1)} + h^{(n-1)} W^{(n)} h^{(n)}\right), \quad (1.1)$$

$$p\left(h_i^{(k)} = 1 | h^{(k+1)}\right) = \sigma\left(b_i^{(k)} + W_{:,i}^{(k+1)\top} h^{(k+1)}\right) \forall i, \forall k \in 1, \dots, n-2, \quad (1.2)$$

у випадку бінарних входних нейронів:

$$p\left(v_i = 1 | h^{(1)}\right) = \sigma\left(b_i^{(0)} + W_{:,i}^{(1)\top} h^{(1)}\right) \forall i, \quad (1.3)$$

інакше, для неперервних вхідних нейронів:

$$\text{v} \sim \mathcal{N}\left(v; b^{(0)} + W^{(1)\top} h^{(1)}, \beta^{-1}\right), \quad (1.4)$$

де $h^{(n)}$ – стани нейронів n-го прихованого шару, v – випадковий зразок даних, $v$ – згенерований стан видимих нейронів, $\beta^{-1}$ – діагональна матриця, $\sigma(\cdot) -$ сигмоїдальна функція.

На відміну від інших нейромережевих моделей, елементарною ланкою ГМД є не окремий шар, а пара сусідніх шарів, що утворюють обмежену машину Больцмана (ОМБ), інакше кажучи, ГМД утворена каскадом ОМБ. А отже ГМД із одним прихованим шаром вироджується у ОМБ.

Структура ГМД зображена на рис. 1.1.

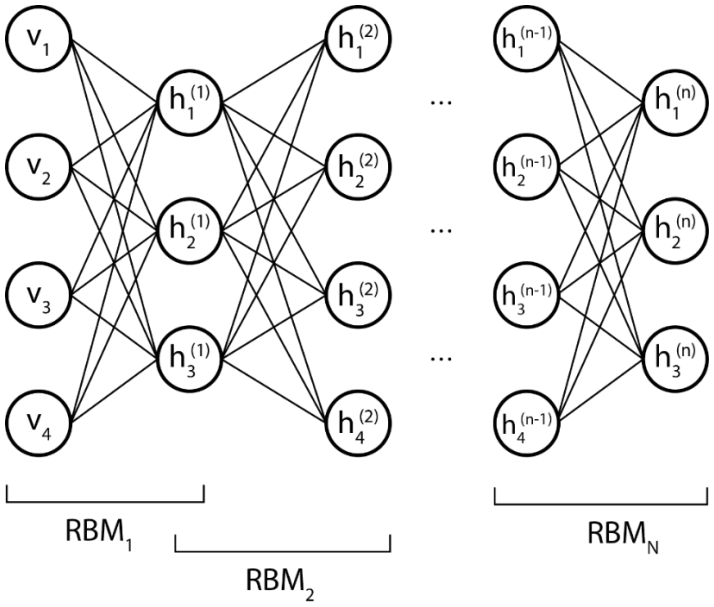



Рис. 1.1. Структура нейронної глибокої мережі глибокої довіри

Робота навченої ГМД полягає у генерації даних шляхом розповсюдження сигналу від останнього прихованого шару до видимого. Для генерації зразку потрібно створити Гіббс-виборку на останній парі шарів, тобто на останній у каскаді ОМБ, далі сигнал розповсюджується мережею до вхідних нейронів, де відбувається формування результуючого вектору.

Навчання ГМД [3] починається з навчання першої ОМБ, яка має максимізувати

$$\mathbb{E}_{v\sim p_{data}} \log p(v) \tag{1.5}$$

за допомогою відповідних алгоритмів, де $\mathbb{E}_{v\sim p_{data}}$– математичне очікування значення вектора видимого шару (першого), $p_{data}$ – розподіл ймовірностей вхідних даних $v$, $p(v)$ – розподіл ймовірностей станів видимого шару $v$.

Отримані параметри ОМБ визначають параметри першого шару ГМД. Далі навчають другу ОМБ максимізувати [3]

$$\mathbb{E}_{v\sim p_{data}} \mathbb{E}_{h^{(1)}\sim p^{(1)}(h^{(1)}|v)} \log p^{(2)}\left(h^{(1)}\right), \tag{1.6}$$

де $p^{(1)}$ – розподіл ймовірності, представленої першою ОМБ, а $p^{(2)}$ – розподіл, представлений другою ОМБ. Тобто друга ОМБ навчається моделювати розподіл, визначений вибіркою із прихованих нейронів першої ОМБ, а перша ОМБ працює прямо із тренувальними даними.

Ця процедура, що має назву жадібного пошарового тренування, продовжується ітеративно до досягнення очікуваного результату, що задовольняє відповідним критеріям ефективності [4].

Навчену ГМД можна безпосередньо використовувати у якості генеративної моделі, проте головним чином її використовують у зв'язку із її здатністю покращувати моделі класифікації. Тобто, можна використовувати ваги із ГМД для ініціалізації багатошарового персептрона (БШП) з переднавчанням [3]:

$$h^{(1)} = \sigma\left(b^{(1)} + v^{\top} W^{(1)}\right), \tag{1.7}$$

$$h^{(l)} = \sigma\left(b^{(l)} + h^{(l-1)\top} W^{(l)}\right) \forall l \in 2, \ldots, m, \tag{1.8}$$

де $m$ – кількість прихованих шарів персептрона.

Після ініціалізації БШП вагами із зміщеннями, навченими на етапі первинного навчання ГМД, БШП донавчають для розв'язання задачі класифікації. Цей процес називають дискримінантнм фінальним донавчанням (fine tunning). Структура такої гібридної мережі наведена на рис. 1.2.

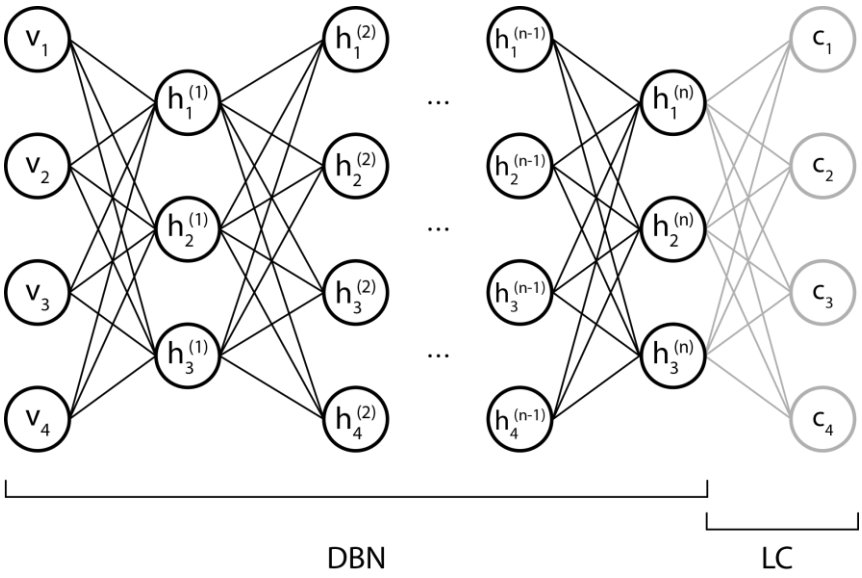



Рис. 1.2. Структура багатошарового персептрона, утвореного із глибокої мережі довіри. DBN – глибока мережа довіри, LC – лінійний класифікатор

Для контролю якості моделі відбувається оцінка її узагальнюючої здатності на тестовому наборі даних.

## 1.2. Глибока мережа довіри та напівкероване навчання

Напівкероване навчання (semi-supervised learning) – метод машинного навчання, що є гібридом методів навчання без вчителя (unsupervised learning) та навчання із вчителем (supervised learning), та використовується для покращення методу навчання із вчителем із використанням нерозмічених тренувальних даних [5]. Задача, що вирішує цей метод, полягає у наступному:

Нехай існує множина зразків даних $x = \{x_1, \dots x_n | x_i \in X\}$ та множина відповідних міток $y = \{y_1, \dots y_n | y_i \in Y\}$. До того ж існує додактово множина зразків даних $x = \{x_{n+1}, \dots x_{n+m} | x_i \in X\}$, які не мають відповідних ним міток $y$. Використовуючи комбінацію розмічених та нерозмічених даних, потрібно досягти найкращого результату класифікації, порівняно із використанням лише розмічених даних.

Для застосування цього методу повинні виконуватись [6] наступні припущення:

- точки, що лежать близько одна до іншої, розмічені однаково із більшою ймовірністю;
- дані переважно утворюють дискретні кластери, точки із одного кластера розмічені однаково із більшою ймовірністю;
- дані надлишкові, тобто утворюють багатоманітність значно меншої розмірності, ніж вхідний простір.

Для напівкерованого навчання із застосуванням великих обсягів даних застосовують генеративні нейромережеві моделі, представником яких є глибока мережа довіри.

Зважаючи на властивості ГМД, ціль навчання такої моделі можна виразити наступним чином [6]:

$$argmax_\theta(\log p(\{x_i, y_i\}_{i=1}^n | \theta) + \lambda \log p(\{x_i\}_{i+n=1}^{n+m} | \theta)), \tag{1.9}$$

де $\theta$ – сукупність параметрів мережі ($W$ та $b$).

Напівкероване навчання має 2 варіанти використання [5][6]:

1) для тренування моделі доступний великий обсяг розмічених даних;

2) для створення та тренування моделі доступний великий обсяг нерозмічених даних та значно менший обсяг розмічених.

Процедура напівкерованого навчання містить наступні кроки:

1) Побудова та навчання ГМД навчанням без вчителя із використанням повного обсягу тренувальних даних;
2) Додавання до ГМД шару класифікатора;
3) Дискримінативне донавчання мережі із використанням розміченої частини тренувального набору із контролем якості моделі.

### 1.3. Підходи до побудови глибоких мереж довіри

Вибір структури глибоких штучних нейронних мереж й досі є актуальною проблемою, що досліджується вченими у галузі машинного навчання по усьому світі [3]. Через відсутність єдиної теорії, що описувала б цю проблему, та давала б єдиний алгоритм її розв’язання, дослідники та інженери змушені створювати та використовувати різноманітні еврістичні методики побудови глибоких нейромережевих моделей, починаючи із ручного задання їх параметрів і до створення програмних продуктів, що виконують автоматизований пошук оптимальних структур.

Практикують два підходи до побудови ГМД:

1) Апріорне задання структури мережі;
2) Пошук оптимальної структури ГМД.

Перший підхід застосовують, коли є відомою бажана структура цільової моделі, наприклад БШП, що може бути зумовлено, наприклад її специфікою застосування, обмеженням обчислювальних ресурсів, тощо. При цьому підході відбувається створення каскаду ОМБ із заданими параметрами глибини та ширини та пошарове жадібне навчання. Метою навчання моделі при такому підході є досягнення максимально можливої якості моделі із заданими обмеженнями на структуру.

У другому підході, коли бажана структура цільової моделі невідома, виконується пошук оптимальної структури ГМД, шляхом ітеративного

нарощування каскаду ОМБ. Мета навчання у такому підході полягає у досягненні заданого порогового рівня якості моделі.

Проведені дослідження [7] доводять, що при цьому ширину шарів можна обмежити зверху значенням:

$$w = n + 4, \tag{1.10}$$

де $n$ – розмір вхідного шару мережі, чи, іншими словами, розмір вектора зразка даних із тренувального набору. [4].

Після досягнення глибини мережі, що дає достатню якість її здатності узагальнення, додатково іноді проводять «полегшення» мережі, шляхом видалення надлишкових зв'язків чи нейронів [8].

Задача структурного синтезу глибокої мережі довіри зводиться до ітеративної побудови каскаду обмежених машин Больцмана, із покроковим навчанням їх та валідацією моделі наприкінці кожної ітерації [3].

Критерієм якості під час навчання ГМД є похибка узагальнення на тестовому наборі даних. Для оцінки генеративної якості ГМД застосовується критерій абсолютного відхилення відтвореного та заданого зразків даних:

$$e = \frac{1}{N}\sum_{i=1}^{N}|v_i - \hat{v}_i|, \tag{1.11}$$

де $v$ – тестовий зразок даних, $\hat{v}$ – згенерований зразок, $e$ – похибка відтворення, $N$ – розмір тестового набору даних.

Якщо ж ГМД використовується для переднавчання БШП застосовуються оцінки помилки та точності класифікації на тестовому наборі даних:

$$e = \frac{1}{N}\sum_{i=1}^{N}|c_i - \hat{c}_i|, \tag{1.12}$$

$$a = \frac{g}{N}, \tag{1.13}$$

де $c_i$ – клас тестового зразку даних, $\widehat{c_i}$ – вихід мережі, $e$ – похибка класифікації, $N$ – розмір тестового набору даних, $a$ – точність класифікації, $g$ – кількість правильно класифікованих зразків.

## 1.4. Проблеми створення глибоких мереж довіри

У випадку, коли структура ГМД не була задана апріорно, постає проблема відшукання оптимальної структури та налаштування оптимальних параметрів за такої структури. Оскільки нейронні мережі все ще не мають універсального аналітичного опису із точки зору оптимальної структури для різних задач, застосовуються емпірично обумовлені евристики. Хоча таких прийомів досить багато, їх вибір та застосування найчастіше залежить від користувача, експерта. Проте очевидна необхідність автоматизації пошуку оптимальної структури ГМД, оскільки ручний її підбір та налаштувань навчання є дуже витратним та часто рутинним [3].

Метою роботи є створення програмного засобу, що виконує структурно-параметричний синтез глибокої мережі довіри, тобто реалізує наступні алгоритми:

- Пошуку оптимальної структури ГМД;
- Пошуку оптимальних гіперпараметрів навчання ГМД та виконання навчання;
- Остаточного дискримінантного донавчання ГМД;
- Перетворення ГМД у цільову модель.

Для досягнення мети мають бути виконані наступні кроки:

- Пошук та аналіз доступних та ефективних інструментів створення та тренування глибоких мереж довіри та обмежених машин Больцмана;
- Розробка програмного забезпечення для реалізації процесів побудови та налаштування необхідних нейромережевих моделей; засобів моніторингу процесів під час роботи системи;
- Реалізація кроків прийому, підготовки та збереження наборів даних; а також збереження результатів роботи системи у вигляді програмної моделі, придатної для цільового використання у виробничому середовищі.

Створений продукт повинен відповідати наступним вимогам:

- Реалізувати кроки та алгоритми, що реалізують мету даної роботи;
- Мати зручний та документований інтерфейс користувача;

- Надавати користувачеві можливість отримати поточні результати роботи системи;
- Реалізувати можливість повної зупинки роботи системи, та відновлення роботи від точки зупинки із збереженням попередніх результатів, або повного рестарту процесу із видаленням попередніх результатів;
- Має бути достатньо простим у налагодженні та розгортанні у виробничому середовищі.

## 1.5. Огляд існуючих рішень

Проблема автоматизації побудови та навчання оптимальної моделі глибокої нейронної мережі існує давно, а отже проведено досить багато досліджень у пошуках її розв’язання [3].

На сьогодні існує як мінімум два підходи, що напрямлені на полегшення побудови та тренування ефективних моделей експертом:

1. Автоматизація окремих кроків побудови та навчання моделі;
2. Перевикористання існуючих налаштованих моделей загального використання із підлаштуванням під застосування для конкретної задачі. Цей підхід має назву «передача знань» («transfer learning»), має свою специфіку, переваги та недоліки та детально описаний у наступному розділі.

У напрямку автоматизації побудови та навчання нейронних мереж існують певні наукові та інженерні напрацювання, що реалізувались у формі спеціалізованих програмних бібліотек та фреймворків.

### 1.5.1. Google Cloud AutoML

Цей інструмент для побудови широкого спектра моделей для цілої низки задач машинного навчання розроблений підрозділом Cloud AI компанії Google та реалізований у формі хмарного онлайн сервісу, який дозволяє автоматично будувати нейромережеві моделі, тренувати їх та далі використовувати у складі інших програмних систем, побудованих у екосистемі хмарних сервісів цієї компанії.

Серед переваг цього інструменту можна назвати:

- Виконує побудову гіпотез на основі вхідних даних без втручання користувача;
- Не потребує кваліфікації користувача в області машинного навчання, простота використання;
- Результат може бути легко вбудований в іншу програмну систему, що була розгорнута у тій же інформаційній екосистемі хмарних сервісів Google.

Недоліки:

- «Чорна скринька» – відсутній контроль за процесом побудови та навчання моделі; немає можливості оцінити проміжні результати роботи тощо;
- Увесь обсяг вхідних даних має бути розмічений;
- Незручність використання отриманих результатів поза хмарними онлайн сервісами Google.

### 1.5.2. H2O.ai

Фреймворк, створений для автоматизації операцій побудови та тренування різноманітних моделей машинного навчання, включно із глибокими нейронними мережами. Він призначений для досвічених кваліфікованих спеціалістів та дослідників машинного навчання, пропонує широкий спектр моделей, дозволяє керувати процесом, здатний до створення моделей придатних для розгортання у виробничому середовищі різних типів.

Переваги:

- Широкий спектр моделей;
- Розвинена система конфігурації;
- Доступний контроль за процесом побудови та навчання моделі; дає можливості користувачеві оцінити проміжні результати роботи та втрутитись для переналаштування тощо;
- Здатність до інтерпретації отриманих результатів;

- Відносно легко розгортається у різного рівня та типу середовищі (від локальної робочої станції, локального сервера, обчислювального кластера й до хмарних сервісів), завдяки розвиненому набору інтерфейсів та використанню технологій контейнеризації та оркестрації;
- Пристосований для роботи із фреймворками обробки великих даних, розподілених обчислень;
- Дружній до хмарних обчислювальних сервісів Amazon, Google, Microsoft.

Недоліки:

- Високий «поріг входу», занадто складний інструмент для відносно невеликих та простих задач;
- Для створення, тренування та перетворення глибоких мереж довіри потребує допрацювання;
- Для роботи із гібридним (нерозмічена+розмічена частини) набором даних потребує складної конфігурації.

### 1.5.3. TPOT

Фреймворк, що позиціонується авторами як асистент для дослідників даних. Також створений для виконання повного циклу створення моделі машинного навчання: від попередньої обробки даних, через пошук моделі до тренування обраного варіанту. Базується на генетичних алгоритмах пошуку оптимальних архітектур моделей та параметрів їх навчання.

Переваги:

- Зручний для дослідників, гнучкий у конфігурації та використанні;
- Прозорість процесів для моніторингу.

Недоліки:

- Складність використання розподілених обчислень без додаткового програмування;
- Мала придатність до використання із великими обсягами даних, більше підходить для дослідження окремих моделей на малих датасетах;

- Складність використання із інструментами обчислення на графічних процесорах, відсутність відповідних налаштувань, необхідність створення додаткових інтерфейсів;
- Необхідність післяобробки результатів роботи для застосування у реальному виробництві.

### 1.5.4. Scikit-learn

Популярний фреймворк класичного машинного навчання, що містить величезний набір інструментів обробки даних, моделей машинного навчання, багату документацію та безліч прихильників. Дуже поширений серед спеціалістів у статистиці, дослідників та студентів. Також містить інструменти для автоматичного пошуку структури моделей машинного навчання та гіперпараметрів їх навчання.

Переваги:

- Великий набір іструментів, багата документація;
- Можливість автоматичного пошуку структур моделей та параметрів навчання.

Недоліки:

- Відсутність зручних інструментів побудови та навчання глибоких нейронних мереж;
- Відсутність можливості застосування обчислень на графічних процесорах без додаткового програмного забезпечення;
- Необхідність післяобробки результатів роботи для застосування у реальному виробництві.

### Висновки по розділу

У цьому розділі проведено аналіз предметної області та існуючих рішень, проаналізовані основні сучасні розробки, визначено переваги та недоліки цих рішень. В результаті проведеного аналізу сформульована постановка задачі, наведене призначення, цілі та задачі розробки.

# РОЗДІЛ 2. ІЄРАРХІЧНА СХЕМА СТРУКТУРНО-ПАРАМЕТРИЧНОГО СИНТЕЗУ ГЛИБОКИХ МЕРЕЖ ДОВІРИ

## 2.1. Стратегія структурно-параметричного синтезу глибоких мереж довіри

Глибока мережа довіри структурно є каскадом обмежених машин Больцмана. А отже задача побудови ГМД полягає у поступовому додаванні ОМБ, починаючи із першої, що приймає вхідні дані.

Стратегія синтезу ГМД має наступні кроки:

1) Ініціалізація ГМД;
2) Додавання ОМБ до існуючої структури. Якщо це перша ланка каскаду, то вхідні дані формуються безпосередньо із тренувального датасета, інакше, вхідні дані формуються проходженням даних тренувального датасета крізь попередні ОМБ у каскаді;
3) Тренування поточної ОМБ;
4) Додавання шару класифікатора та дискримінантне донавчання ГМД;
5) Перевірка усієї ГМД на якість за критерієм мінімізації похибки узагальнення (точність класифікації) на тестовому наборі даних;
6) Якщо похибка узагальнення перевищує цільову, шар класифікатора видаляється, існуючі параметри мережі фіксуються та повторюються кроки 2–6;

Після досягнення необхідної якості ГМД процес структурно-параметричного синтезу вважається закінченим, а модель – побудованою та навченою.

Для валідації результатів навчання ОМБ використовується подвійна перевірка генеративної (за величиною похибки відтворення тестового зразка) та класифікаційної (за точністю класифікації на тестовому наборі розмічених даних) здатності моделі. У випадку використання ГМД для навчання на нерозміченому наборі даних виконується лише валідація на генеративну здатність мережі.

Параметри ГМД таким чином пошарово повторюють параметри відповідної ОМБ, а отже головна задача налаштування параметрів ГМД перетворюється у покрокове навчання кожної ОМБ у каскаді.

## 2.2. Обмежена машина Больцмана, її структура та визначення

Обмежена машина Больцмана – генеративна стохастична енергетична штучна нейронна мережа, що здатна вивчити розподіл ймовірностей у наборі вхідних даних. Вона є розвитком універсальної машини Больцмана (УМБ) та була створена для уникнення практичних обмежень УМБ, наприклад, практичної неефективності виборки із розподілу, обчслення значення енергії тощо. Для розв’язання цих проблем на УМБ були накладені структурні обмеження, що значно спростило її навчання та використання [2].

Як походить із назви, ОМБ є варіацією машини Больцмана із забороною на зв’язки між нейронами одного шару, що перетворює модель на дводольний граф, у якому одна доля відповідає шару «видимих» нейронів $v_i$, а друга – «прихованих» $h_i$. Структура ОМБ наведена на рис. 2.1.

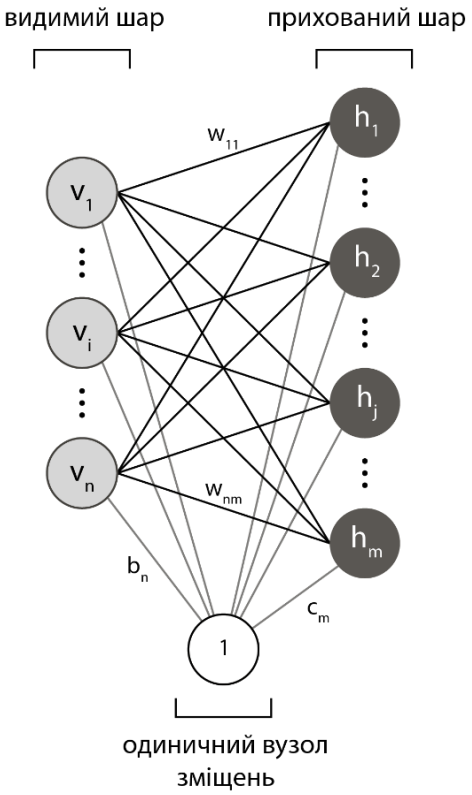



Рис. 2.1. Структура ОМБ

Обмежена машина Больцмана – енергетична модель, у якій сумісний розподіл ймовірності описується функцією енергії. Для канонічної ОМБ із бінарними видимими та прихованими нейронами [3]:

$$p(v,h) = \frac{1}{Z}\exp\big(-E(v,h)\big), \tag{2.1}$$

$$E(v,h) = -b^\top v - c^\top h - v^\top W h, \tag{2.2}$$

$$Z = \sum_v \sum_h \exp\big(-E(v,h)\big), \tag{2.3}$$

де $Z$ – нормалізатор або функція розбиття, або статистична сума, $E$ – повна енергія системи, $v$ та $h$ – вектори стану видимих та прихованих нейронів, $b$ та $c$ – вектори зміщення видимих та прихованих нейронів, $W$ – матриця зв'язків між нейронами сусідніх шарів.

Із визначення функції нормалізатора $Z$ очевидно, що вона не може бути ефективно обчислена для довільної моделі та довільного набору даних, що формально доведено [3]. Це також означає, що сумісний розподіл $p(v,h)$ також не може бути обчислений.

Хоча $p(v,h)$ не може бути знайдений, ОМБ як дводольний граф має властивості умовних розподілів $p(v|h)$ та $p(h|v)$, які є факторними та вже можуть бути відносно легко обчислені та вибрані.

Умовні розподіли можна вивести із сумісного розподілу:

$$p(h|v) = \frac{p(h,v)}{p(v)}, \tag{2.4}$$

$$p(h|v) = \frac{1}{p(v)}\frac{1}{Z}\exp\big(b^\top v + c^\top h + v^\top W h\big), \tag{2.5}$$

$$p(h|v) = \frac{1}{Z}\exp\big(b^\top v + c^\top h + v^\top W h\big), \tag{2.6}$$

$$p(h|v) = \frac{1}{Z'}\exp\big(c^\top h + v^\top W h\big), \tag{2.7}$$

$$p(h|v) = \frac{1}{Z'} \exp\left(\sum_{j=1}^{n_h} c_j h_j + \sum_{j=1}^{n_h} v^\top W_{:,j} h_j\right), \tag{2.8}$$

$$p(h|v) = \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\left(c_j h_j + v^\top W_{:,j} h_j\right), \tag{2.9}$$

де $n_h$ – розмір прихованого шару.

Оскільки в умовній частині знаходяться видимий шар $v$, то можно розглядати його як постійний відносно розподілу $p(h|v)$. Факторна природа умовного розподілу $p(h|v)$ дає можливість виразити сумісний розподіл вектора $h$ у вигляді добутку ненормованих розподілів окремих елементів $h_j$.

Нормовані розподіли індивідуальних нейронів $h_j$ матимуть вигляд

$$p\left(h_j = 1|v\right) = \frac{\tilde{p}\left(h_j = 1|v\right)}{\tilde{p}\left(h_j = 0|v\right) + \tilde{p}\left(h_j = 1|v\right)}, \tag{2.10}$$

$$p\left(h_j = 1|v\right) = \frac{\exp\left(c_j + v^\top W_{:,j}\right)}{\exp(0) + \exp\left(c_j + v^\top W_{:,j}\right)}, \tag{2.11}$$

$$p\left(h_j = 1|v\right) = \sigma\left(c_j + v^\top W_{:,j}\right) \tag{2.12}$$

Тоді повний умовний розподіл можна виразити у вигляді добутку розподілів індивідувальних нейронів:

$$p(h|v) = \prod_{j=1}^{n_h} p\left(h_j|v\right) \tag{2.13}$$

Таким же чином можна вивести умовний розподіл для нейронів видимого шару

$$p(v|h) = \prod_{j=1}^{m_v} p(v_i|h), \tag{2.14}$$

де $m_v$ – розмір видимого шару.

Хоча канонічна ОМБ містила лише бінарні нейрони [3], що означає, що вона може бути застосована лише із бінаризованими даними, існує розвиток ОМБ для неперервних вхідних даних, у якому шар видимий шар втілює функцію

нормального розподілу, а приховані – функцію Бернулі. Такая ОМБ називається ОМБ Гауса-Бернулі (Gauss-Bernoulli RBM, GBRBM), яка має інший вираз умовної ймовірності та функції енергії:

$$p(v|h) = \mathcal{N}(v; Wh, \beta^{-1}), \tag{2.15}$$

$$E(v,h) = \frac{1}{2} v^{\top}(\beta \odot v) - (v \odot \beta)^{\top} Wh - b^{\top} h, \tag{2.16}$$

де $\beta$ – матриця точності, або у термінах дисперсії

$$p(v|h) = \mathcal{N}(v; b, \sigma), \tag{2.17}$$

$$E(v,h) = \sum_{i=1}^{n} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j}^{m} c_j h_j - \sum_{i,j} \frac{v_i h_j w_{ij}}{\sigma_i}, \tag{2.18}$$

де $b$ – середнє значення (математичне очікування), а $\sigma$ – середньоквадратичне відхилення.

## 2.3. Навчання обмеженої машини Больцмана

Обмежена машина Больцмана використовує навчання без вчителя (unsupervised learning). Головною метою навчання є знаходження невідомих прихованих (латентних) характеристик у вхідних даних, шляхом двонапрямленого проходження сигналу, і отримання виходу мережі максимально близького до вхідного образу. Цей аспект робить обмежену машину Больцмана східну за призначенням до автоенкодера [3][9].

Результатом навчання обмеженої машини Больцмана є налагоджені параметри синаптичних зв’язків між шарами, а також зміщення прихованого шару, які і є відображенням шуканих прихованих характеристик, розподілених у тренувальних даних.

Стандартний шлях для оцінки параметрів статистичних моделей є оцінка максимуму правдоподібності.

Цільовою функцією навчання мережі є максимізація ймовірності відтворення вхідного вектора:

$$P(V) = \prod_{v \in V} p(v), \tag{2.19}$$

$$\theta = \arg\max_{\theta} \prod_{v \in V} p(v), \tag{2.20}$$

де $V$ – множина зразків тренувального набору даних, $\theta$ – шукані параметри, тобто ваги синаптичних зв'язків між видимим і прихованим шаром, включно із зміщенням.

Оскільки логарифм неперервної функції зростає подібно до зростання самої функції, а добуток значно складніше обчислювати, ніж суму, то цільову функцію доречно визначити як логарифм ймовірності:

$$\begin{aligned} \ln \mathcal{L}(\theta|v) = \ln p(v|\theta) = \ln \frac{1}{Z} \sum_{h} e^{-E(v,h)} = \\ = \ln \sum_{h} e^{-E(v,h)} - \ln \sum_{v,h} e^{-E(v,h)} \end{aligned} \tag{2.21}$$

Градієнт по параметрах:

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} = \frac{\partial}{\partial \theta}\left( \ln \sum_{h} e^{-E(v,h)} \right) - \frac{\partial}{\partial \theta}\left( \ln \sum_{v,h} e^{-E(v,h)} \right), \tag{2.22}$$

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} = -\sum_{h} p(h|v) \frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta} \tag{2.23}$$

Оскільки нормалізатор $Z$ (який неявно виражений у другому члені останнього виразу) неможливо ефективно порахувати навіть при відносно невеликих розмірах мережі, через експоненційне зростання складності, застосовують апроксимацію, яка реалізована через метод навчання обмеженої машини Больцмана, який дозволяє уникнути обчислення Z, який був розроблений Дж. Хінтоном, і який має назву порівняльного розходження (Contrastive Divergence, CD) [9].

### 2.3.1. Contrastive Divergence

Складність виборки у ОМБ полягає у тому, що вона є неорієнтованою графічною моделлю, а отже складно визначити $p(v|h)$ та $p(h|v)$ у конкретний момент часу без задання порядку виборки. Розв'язання цієї проблеми базується методі Монте Карло із Марковським Ланцюгом (Monte Carlo Markov Chain,

МСМС), який дозволяє виконати виборку із розподілу, близького до оригінального за умови довжини ланцюга, що прямує до нескінченності.

Марковський ланцюг є дискретним у часі стохастичним процесом, для якого зберігаються Марковські властивості, а саме: кожен наступний стан залежить лише від попереднього стану, і не залежить від усіх інших попередніх. Метою застосування МСМС для ОМБ є отримання такого розподілу марківського ланцюга $q(x)$, що відповідатиме розподілу ОМБ $p_{model}(x)$.

Отримання незміщених оцінок для логарифму правдоподібності із використанням метода МСМС зазвичай потребує невизначено великої кількості кроків для досягнення рівноважного розподілу, що має назву час перемішування. Перевірка факту досягнення марківським ланцюгом рівноваги також є важким завданням.

Було доведено, що оцінки отримані після запуску ланцюга лише з кількох кроків може бути достатнім для тренування моделі, якщо після кожної ітерації уточнювати розподіл ймовірності, через оновлення параметрів моделі. Це веде до алгоритму CD [9], який зараз є основою для тренування ОМБ [10].

У якості алгоритма виборки із МСМС використовується Гіббс-виборка, що є дуже простою у реалізації та обчислювально ефективною. Його перевагою є те, що для нього не потрібний сумісний розподіл, а достатньо умовних розподілів для кожної змінної.

У канонічному варіанті ОМБ із бінарними видимими та прихованими нейронами у якості функції активації нейронів застосовується сигмоїдна функція вигляду:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.24}$$

Умовні ймовірністі прихованих і видимих нейронів мають вигляд:

$$p(h_j = 1|v) = \sigma\left(c_j + \sum_i v_i\, w_{ij}\right), \tag{2.25}$$

$$p(v_i = 1|h) = \sigma\left(b_j + \sum_j h_j\, w_{ij}\right) \tag{2.26}$$

Тоді можна знайти часткові похідні по вагах:

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{original} - \langle v_i h_j \rangle_{model}, \tag{2.27}$$

де $\langle \cdot \rangle_{original}$ та $\langle \cdot \rangle_{model}$ – математичне очікування оригінального та відтвореного зразка вхідних даних, у відповідності до розподілу ймовірностей.

А поправки параметрів легко відшукуються:

$$\begin{aligned} \Delta w_{ij} &= \varepsilon\big(\langle v_i h_j \rangle_{original} - \langle v_i h_j \rangle_{model}\big), \\ \Delta b_{ij} &= \varepsilon\big(\langle v_i \rangle_{original} - \langle v_i \rangle_{model}\big), \\ \Delta c_{ij} &= \varepsilon\big(\langle h_j \rangle_{original} - \langle h_j \rangle_{model}\big), \end{aligned} \tag{2.28}$$

де $\varepsilon$ – швидкість навчання.

Ідея $k$-крокового CD алгоритма полягає у наступному: замість апроксимації другого члена градієнта логарифма правдоподібності виборки із ОМБ-розподілу (що потребує ресурсоємного обчислення марківського ланцюга, доки не буде досягнуто стаціонарного розподілу), Гіббс-ланцюг виконується лише k кроків (зазвичай навіть $k = 1$). Гіббс-ланцюг ініціалізується зразком тренувальних даних $v^{(0)}$ та закінчується виборкою $v^{(k)}$ після k кроків. Кожен крок $t$ складається із виборки стану прихованого шару $h^{(t)}$ з ймовірності $p(h|v^{(t)})$ та наступної виборки $v^{(t+1)}$ з $p(v|h^{(t)})$. Градієнт по відношенню до параметрів $\theta$ функції максимальної правдоподібності таким чином має вигляд:

$$\frac{\partial \log \mathcal{L}\big(\theta \big| v^{(0)}\big)}{\partial \theta} = -\sum_h p\big(h\big|v^{(0)}\big)\frac{\partial E\big(v^{(0)}\big|h\big)}{\partial \theta} + \sum_{v,h} p\big(v^{(k)}\big|h\big)\frac{\partial E\big(v^{(k)}\big|h\big)}{\partial \theta} \tag{2.29}$$

Виборка зразків у алгортимі CDk виглядає наступним чином:

1) Стан видимих нейронів приймає значення вхідного зразка;
2) Отримуються ймовірності станів прихованих нейронів;

3) Для кожного прихованого нейрона відбувається виборка на основі отриманої ймовірності;
4) Сигнал проходить від прихованого шару до видимого і отримуються ймовірності станів видимих нейронів;
5) Для кожного видимого нейрона відбувається виборка на основі отриманої для нього ймовірності;
6) Якщо номер ітерації менше k, повертаються до кроку 2;
7) Отримуються ймовірності станів прихованих нейронів.

Графічно цей процес зображено на рис. 2.2.

Оновлення параметрів відбувається за формулами (2.28).

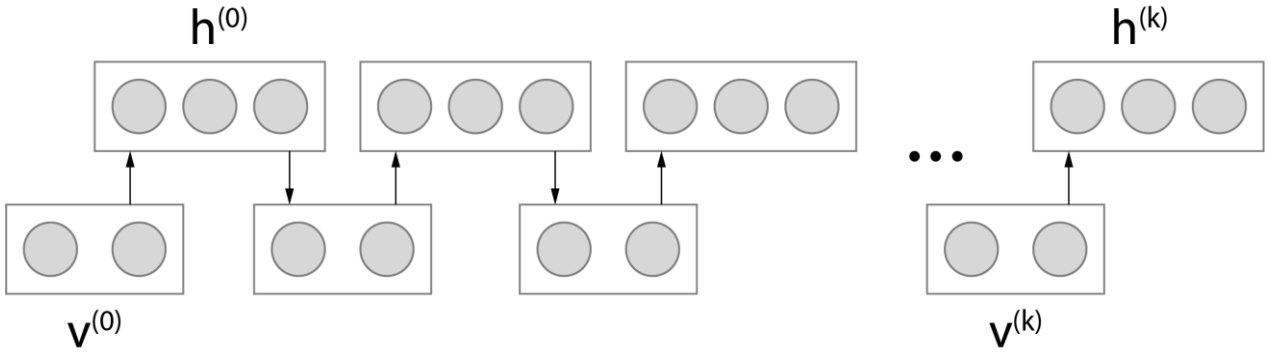



Рис. 2.2. Марковський ланцюг, що втілює вибіркову частину алгортима CDk

Таким чином чим довше відбувається виборка (чим довший ланцюг), тим точнішими стають градієнти. У той же час Хінтон [9] показав, що навіть довжина ланцюга $k = 1$ може бути достатньою у разі використання достатньо великої кількості навчальних ітерацій або епох. Перша частина кроку називається позитивною фазою, а друга частина – негативною фазою.

У Гіббс-виборці перша частина у виразі градієнта характеризує розподіл у даних у момент часу $t = 0$, а друга – реконструйований або згенерований стан у момент $t = k$. Виходячи із цього CDk процедура може бути описана як:

$$v(0) \to h(0) \to v(1) \to h(1) \to \cdots \to v(k) \to h(k) \qquad (2.30)$$

Результатом є наступні правила навчання мережі ОМБ з огляду на метод градієнтного спуска:

$$w_{ij}(t+1) = w_{ij}(t) + \varepsilon \frac{\partial lnP(x)}{\partial w_{ij}(t)}, \tag{2.31}$$

для постійного навчання:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \varepsilon \Big( x_i(0) y_j(0) - x_i(k) y_j(k) \Big), \\ b_i(t+1) &= b_i(t) + \varepsilon \big( x_i(0) - x_i(k) \big), \\ c_j(t+1) &= c_j(t) + \varepsilon \Big( y_j(0) - y_j(k) \Big) \end{aligned} \tag{2.32}$$

Для пакетного (batch) навчання формули приймають вигляд:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \varepsilon \sum_{l=1}^{L} \Big( x_i^l(0) y_j^l(0) - x_i^l(k) y_j^l(k) \Big), \\ b_i(t+1) &= b_i(t) + \varepsilon \sum_{l=1}^{L} \Big( x_i^l(0) - x_i^l(k) \Big), \\ c_j(t+1) &= c_j(t) + \varepsilon \sum_{l=1}^{L} \Big( y_j^l(0) - y_j^l(k) \Big), \end{aligned} \tag{2.33}$$

де $L$ – розмір пакета.

Як видно із наведених формул, правила навчання ОМБ полягає у мінімізації різниці між оригінальними даними та згенерованими результатами способом Гіббс-виборки [9][11].

### 2.3.2. Persistent Contrastive Divergence

Алгоритм CD-1 досить швидкий, має відносно низьку дисперсію та дає прийнятну апроксимацію градієнта правдоподібності, та все ще досить значно далеко від шуканого градієнта правдоподібності, коли крок навчання малий. Якщо доступні обчислювальні та часові ресурси дозволяють, то CDk алгоритм за умови $k > 1$ є теоретично більш прийнятним.

Основний алгоритм CD-1 (та й CD-k) полягає у тому, що MCMC ланцюг стартує із зразка даних, отримує відтворений (згенерований) зразок, оновлює параметри моделі, та знов починає роботу із зразка даних.

Оскільки між оновленнями параметрів модель змінюється дуже повільно, є сенс ініціалізації MCMC ланцюга останніми значеннями із попереднього кроку, замість ініціалізації із вхідних даних. Така ініціалізація часто є доволі близькою до розподілу у моделі, навіть при тому, що ваги були трошки оновлені на попередньому кроці, особливо за малого значення кроку навчання. Таким чином у новій ітерації алгоритм наче продовжує MCMC ланцюг у часі, отримуючи перевагу, що надає довгий MCMC ланцюг ($k \gg 1$), та який наближується до шуканого розподілу (при $k \to \infty$). Звичайно, це не той самий довгий ланцюг, оскільки після кожної епохи модель оновлюється, але при достатньо малому кроці навчання це оновлення не має значного впливу на увесь ланцюг, та результатом є отримання достатньо близької апроксимації заданого набору даних [12].

Метод отримав назву постійне контрастивне розходження (Persistent Contrastive Divergence), яке підкреслює, що Марковський ланцюг не реініціалізується із кожною навчальною епохою наново, проте наслідує стан попереднього кроку.

Пізніше цей алгоритм отримав розвиток у так званий швидкий PCD (fast PCD, FPCD) [13]. FPCD намагається досягти швидкого проходження Гіббс-ланцюга шляхом додавання додаткових параметрів $w_{ij}^f, b_j^f, c_j^f$ (для $i = \overline{1,n}$ та $j = \overline{1,m}$), які мають назву швидких параметрів. Цей новий набір параметрів використовується лише для виборки зразка у ланцюзі, але не для застосування у моделі. У процесі обчислення умовних розподілів у Гіббс-виборці, звичайні параметри замінюються сумою швидких та звичайних параметрів, тобто Гіббс-виборка відбувається на основі наступних ймовірностей:

$$\tilde{p}(h_i = 1|v) = \sigma\left(\sum_{j=1}^{m}\left(w_{ij} + w_{ij}^f\right)v_j + \left(c_i + c_i^f\right)\right), \tag{2.34}$$

$$\tilde{p}(v_j = 1|h) = \sigma\left(\sum_{i=1}^{n}(w_{ij} + w_{ij}^f)h_i + (b_j + c_j^f)\right) \quad (2.35)$$

замість умовних ймовірностей, що наведено у виразах (2.25) та (2.26).

Правила оновлення параметрів залишаються такими ж, як і для базового алгоритму, проте крок навчання використовується значно більший, схожий із таким, що застосовується для алгоритма CDk, а також із великим значенням гіперпараметра пригнічення вагів (weight decay).

Після низки досліджень отримано висновок, що ні PCD ані FPCD алгоритми не надають значно кращих результатів, порівняно із CDk.

### 2.3.3. Паралельний відпуск

З метою пошуку кращих результатів навчання ОМБ порівняно із алгоритмом CDk (та його варіантом PCD) було розроблено принципово інший підхід до навчання моделі, що базується на модифікованому методі Монте Карло, що має назву паралельного відпуску (Parallel Tempering, PT) [14].

Основна проблема, що не була розв’язана за допомогою алгоритма CDk полягає у тому, що зразки відтворені у негативній фазі роботи алгоритма не можуть описати загальний стан об’єкта, що моделюється. Отже було запропоновано інший метод, що використовує інший спосіб виборки даних у ланцюзі МСМС.

PT виборка була запропонована у 1980-х роках, коли Свендсен та Ванг (1986) [15] представили алгоритм реплікації МСМС та застосували його до моделі Ісінга, що є еквівалентною до ОМБ із видими нейронами. Симуляція реплікації МСМС полягає у моделюванні кількох реплік за різних температур одночасно, замість генерації їх послідовно. Схожим чином Гейер пізніше застосував паралельна виборка у МСМС ланцюзі для максимізації правдоподібності.

Основна ідея метода PT полягає у Гіббс-виборці із різними температурами, від найвищої при $T = 0$ до низької $T = 1$. Поняття температури у даному контексті означає енергетичний рівень системи. Чим більша температура, тим більша варіативність зразків.

Для кожної пари зразків отриманих при виборці із двох різних ланцюгів обчислюють ймовірність їх перестановки, та після визначення ймовірності відбувається випадкова перестановка, або реплічний обмін на основі правила Метрополіса-Гастінгса:

$$P_{swap}(x_{T_1}, x_{T_2}) = min\left(1, \frac{P_{T_1}(x_{T_2})P_{T_1}(x_{T_1})}{P_{T_1}(x_{T_1})P_{T_2}(x_{T_2})}\right), \tag{2.36}$$

де $T_1$ та $T_2$ визначають температуру двох ланцюгів, $x_{T_1}$ та $x_{T_2}$ – зразки отримані у цих ланцюгах відповідно, $P_{swap}$ – ймовірність перестановки.

Оскільки у даному методі використовується Гіббс-виборка, вираз (2.36) спрощується до наступного [16]:

$$P_{swap}(x_{T_1}, x_{T_2}) = exp\left((T_1 - T_2) \cdot \left(E(x_{T_1}) - E(x_{T_2})\right)\right) \tag{2.38}$$

Після кожної ітерації виборки та перестановки отриманий результат, що відповідає найменшій температурі стає кінцевим для ітерації. Зразки, що отримані із дісного розподілу $P(v, h|\theta)$ проходять достатньо ітерацій, щоб мінімізувати ефект від ініціалізації.

Необхідно зазначити, що Гіббс-ланцюг за найвищої температури $T = 0$ не є мультимодальним, та його нейрони взаємно незалежні та мають ймовірність активації $\frac{1}{2}$. Отже зразки у ланцюзі із високою ймовірністю не уникнуть жодної моди, та не залишаться лише у одній, необов’язково оптимальній.

Як РТ виборка допомагає уникнути одного оптимуму зображено на рис. 2.3.

Природа реплічного обміну між різними температурами допомагає застосувати більш якісну виборку із розподілів поза межами тренувальних даних, та із різних оптимумів цільової функції.

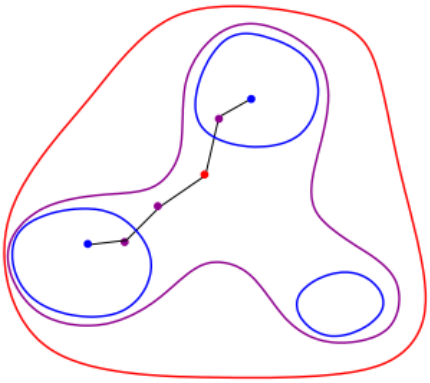


Рис. 2.3. Ілюстрація уникання PT алгоритмом локальних оптимумів. Червоні, фіолетові та сині лінії та точки відображають розподіл та виборку зразків із розподілів із високою, середньою та низькою температурами відповідно. Чорні лінії показують один крок виборки

PT виборка при тренуванні ОМБ може бути застосована для заміни негативної фази у Гіббс-ланцюзі [15][16]. Відповідно із описаними характеристиками, можна очікувати, що отримані зразки будуть описувати розподіл моделі краще, та навчальний процес буде більш ефективним у випадку меншої кількості тренувальних даних, ніж необхідних для алгоритма CDk.

Хоча, потенційно алгоритм PT виборки для ОМБ має переваги над алгоритами сімейства *CD, однак у самій природі PT виборки прихований й головний його недолік, а саме необхідність великої кількості важких обчислювальних операцій для отримання кожного зразка, чого вдало уникають *CD алгоритми.

Зазвичай різні об’єкти та різні набори даних потребують індивідуальних моделей, як за структурою так і за методами їх навчання та параметрами. Цей підхід також стосується й обмеженої машини Больцмана, для якого були розроблені різні алгоритми навчання, що задовольняють різним умовам моделювання. Для зменшення навантаження на експерта, що створює модель, постає задача автоматизації пошуку оптимального алгоритма та гіперпараметрів навчання, що найбільш відповідає тренувальним даним та критеріям ефективності цієї моделі.

Для розв’язання проблеми автоматичного пошуку оптимального алгоритму навчання ОМБ можна використати алгоритм пошуку у просторі варіантів, що зазвичай застосовують для визначення нейромережевої моделі, структурного її синтезу та алгоритма навчання. Доречно застосувати той же підхід й до ОМБ.

## 2.4. Вплив вибору базового алгоритма на навчання обмеженої машини Больцмана

Оскільки сьогодні поширені три алгоритми її навчання, то доречно застосувати алгоритм пошуку по решітці (grid search) (через невелику потужність простору варіантів), що реалізує наступні кроки:

1) Формування множини алгоритмів $alg = \{\text{CD}k, \text{PCD}k, \text{PT}\}$;
2) Формування множини варіантів гіперпараметрів для кожного алгоритма, наприклад, для CD$k$ та PCD$k$ це глибина Марківського ланцюга $k = \{x | 1 \leq x \leq 100\}$. Для алгоритма PT це кількість кроків виборки $k$, що відповідають різним температурам. Автори алгоритма PT показали, що оптимальна його продуктивність досягається при $k$ близькому до розміру тренувального пакету зразків вхідних даних.
3) Формування решітки пошуку комбінуванням множин $alg$ та $k$. Для поточного прикладу розмір решітки дорівнюватиме $|alg| \cdot |k|$.
4) Запуск процесу тренування моделі для кожного варіанта із решітки пошуку.
5) Валідація навчених моделей та вибір найкращого варіанту за критеріями оптимальності на тестовому наборі даних.

За умови наявності достатніх обчислювальних ресурсів пошук по решітці можна провести параллельно, а отже значно скоротити процес знаходження та побудови оптимальної моделі.

Для оцінки впливу того чи іншого алгоритма навчання на сходимість обмеженої машини Больцмана було проведено низку експериментів із застосуванням пошуку по решітці, що утворюється комбінацією параметрів,

наведених у табл. 2.1. Валідація кожного варіанту відбувалась на тестовому наборі даних, у двох режимах: генерація та класифікація.

Структура тестової моделі наведена на рис. 2.4.

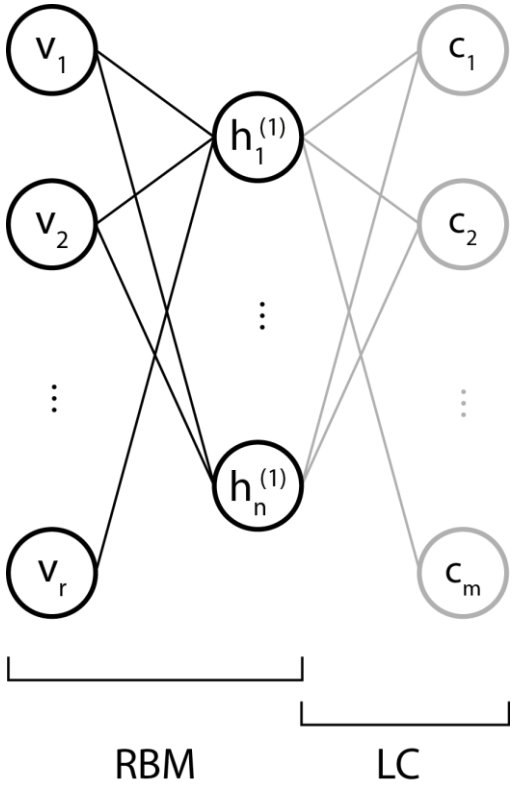



Рис. 2.4. Структура тестової моделі для оцінки генеративної та класифікуючої здатності ОМБ. LC – лінійний класифікатор, $r$ – розмір вхідного шару (відповідає розміру вхідного зразка), $n$ – розмір прихованого шару, $m$ – розмір вихідного шару – лінійного класифікатора (відповідає кількості класів у тестовому наборі)

У режимі генерації валідація вібдувалась із застосуванням критерія похибки відтворення зразків тестового набору на чистій ОМБ.

Для валідації ОМБ у режимі класифікації додатково було виконано наступні кроки:

1) Параметри ОМБ фіксувались;
2) До ОМБ було додано лінійний класифікатор із випадковою ініціалізацією вагів;
3) Ваги лінійного класифікатора додатково навчались.

Валідація ОМБ у режимі класифікації вібдувалась із застосуванням критерія точності класифікації зразків тестового набору.

Таблиця 2.1. Варіанти гіперпараметрів навчання ОМБ для визначення решітки пошуку для проведення експериментів.

| Алгоритм | $k$ |
|---|---|
| CD | [1,2,3,5,15,100] |
| PCD | [1,2,3,5,15,100] |
| PT | [10,100] |

Інші налаштування навчання ОМБ із використанням еталонного датасета MNIST [30] залишаються спільними для усіх варіантів (табл. 2.2).

Таблиця 2.2. Параметри експериментальної моделі ОМБ та гіперпараметри навчання для визначення впливу виду алгортма навчання на результати навчання

| Параметр | Значення |
|---|---|
| Розмір видимого шару | 784 |
| Розмір прихованого шару | 500 |
| Розмір шару класифікатора | 10 |
| Розмір тренувального пакету (batch size) | 100 |
| Кількість навчальних епох | 50 |
| Крок навчання | 0.001 |
| Кількість навчальних епох контрольного класифікатора | 30 |
| Алгоритм оптимізації ОМБ | SGD |
| Розмір тренувального набору даних | 50000 |
| Розмір тестового набору даних | 10000 |

Для алгоритму PT застосування малих значень $k$ не має сенсу, оскільки у такому інтенсивність перемішування (mixing rate) зразків значно падає, отже

алгоритм вироджується у PCD. Для нього має сенс досліджувати варіанти із $k \approx batch\ size$, та контрольний варіант $k = 10$.

Таким чином решітка пошуку містить 14 варіантів, а сама решітка матиме вигляд (табл. 2.3):

Таблиця 2.3. Решітка пошуку оптимального набору гіперпараметрів у експериментах щодо визначення впливу алгоритму навчання ОМБ

| | | | | | |
|---|---|---|---|---|---|
| CD 1 | CD 2 | CD 3 | CD 5 | CD 15 | CD 100 |
| PCD 1 | PCD 2 | PCD 3 | PCD 5 | PCD 15 | PCD 100 |
| PT 10 | PT 100 | - | - | - | - |

Графічні результати експериментів відображені на комбінованих рисунках, які складаються із 2-х частин, що відображають процес навчання у часі (згори) та діаграму результатів навчання після його закінчення (знизу). Нижня частина стовпчиків діаграми відображає фінальну помилку навчання, а верхня – витрати часу на повний цикл навчання. Повна висота стовпчика відображає зважені бали навчання: чим більший бал – тим гірший результат.

### 2.4.1. Алгоритм CD

Результати експериментів наведено у табл. 2.4 та у додатку Г.

Таблиця. 2.4. Результати порівняльних експериментів для алгоритма CDk

| $k$ | Час навчання, сек | Похибка навчання ОМБ | | Точність класифікації |
|---|---|---|---|---|
| | | 1 епоха | 50 епоха | |
| 1 | 59 | 0.062378 | 0.018724 | 0.7855 |
| 2 | 63 | 0.062007 | 0.018274 | 0.7993 |
| 3 | 64 | 0.062221 | 0.018197 | 0.8002 |
| 5 | 76 | 0.062358 | 0.018082 | 0.7787 |
| 15 | 117 | 0.062356 | 0.018077 | 0.7815 |
| 100 | 481 | 0.062376 | 0.018052 | 0.7984 |

Цікавим є дослідження впливу довжини марківського ланцюга $k$ на ефективність ОМБ при застосуванні алгоритма CDk. Хоча, теоретично, збільшення цього параметру покращує сходимість ланцюга, дослідження підтверджують висновки Хінтона [10] про те, що малих значень $k$ достатньо для ефективного навчання ОМБ, оскільки із збільшенням $k$ приріст точності є незначним, порівняно із витратами обчислювальних ресурсів.

У результаті експерименту очевидно, що використання $k > 5$ не є доцільним, та краще збільшити кількість тренувальних епох, а звідси й частоту оновлення параметрів моделі.

### 2.4.2. Алгоритм PCD

Результати експериментів наведені у табл. 2.5 та у додатку Г, свідчать, що значення $k = 1$ для алгоритма PCDk не має сенсу, оскільки у такій версії він стає схожим на алгоритм CDk, а ефект від «зберігання» ланцюга нівелюється щокроковим оновленням параметрів мережі. Найбільшої ефективності алгоритм досягає із значенням $k = 5$. Подальше збільшення цього параметра не дає її приросту, а лише збільшує витрати часу.

Таблиця. 2.5. Результати порівняльних експериментів для алгоритма PCDk

| $k$ | Час навчання, сек | Значення функції втрат | | Точність класифікації |
|---|---|---|---|---|
| | | 1 епоха | 50 епоха | |
| 1 | 54 | 0.199377 | 0.020121 | 0.7675 |
| 2 | 63 | 0.188201 | 0.017520 | 0.7867 |
| 3 | 73 | 0.176479 | 0.017352 | 0.7882 |
| 5 | 76 | 0.154055 | 0.017339 | 0.8035 |
| 15 | 116 | 0.095482 | 0.017591 | 0.7884 |
| 100 | 463 | 0.073712 | 0.017661 | 0.7829 |

### 2.4.3. Алгоритм PT

Результати експериментів наведено у табл. 2.6 та у додатку Г. Недоліки алгоритма PT, порівняно із іншими алгоритмами навчання ОМБ одразу помітні із результатів експериментів. Наприклад, при потужності множини температур $k = 10$, кількість операцій реплікації для отримання одного зразка відтворених даних рівна 10, при цьому для кожної репліки відбувається обчислення енергії моделі, порівняння отриманих значень для усіх сусідніх реплік, обчислення ймовірності реплічного обміну, та виконання операції реплічного обміну.

Уся ця послідовність дій для отримання лише одного зразка даних є значно дорожчою за просту операцію Гіббс-виборки у алгоритмах *CD. Отже, застосування алгоритма PT має сенс лише у випадках, коли алгоритми *CD не дають шуканої ефективності на специфічних наборах даних та при наявності достатньої кількості обчислювальних ресурсів, що дозволять виконувати операції виборки паралельно.

Так, наприклад, для розміру тренувального мінібатча рівного 100 найефективніше використати обчислювальний кластер із 100 обчислювальних машин, які згенерують зразки даних паралельно.

Із обчислювальною системою, що була задіяна для виконання експериментів алгоритм PT використовувати неефективно. Подальша робота на схожій системі відбуватиметься без використання цього алгоритму.

Таблиця. 2.6. Результати порівняльних експериментів для алгоритма PT

| Кількість температур $k$ | Час навчання, хвилин | Значення функції втрат | | Точність класифікації |
|---|---|---|---|---|
| | | 1 епоха | 50 епоха | |
| 10 | 73 | 0.154912 | 0.063076 | 0.7887 |
| 100 | 803 | 0.155582 | 0.059686 | 0.7906 |

Зведені результати усіх експериментів наведені на рис. 2.5 та у додатку Г.

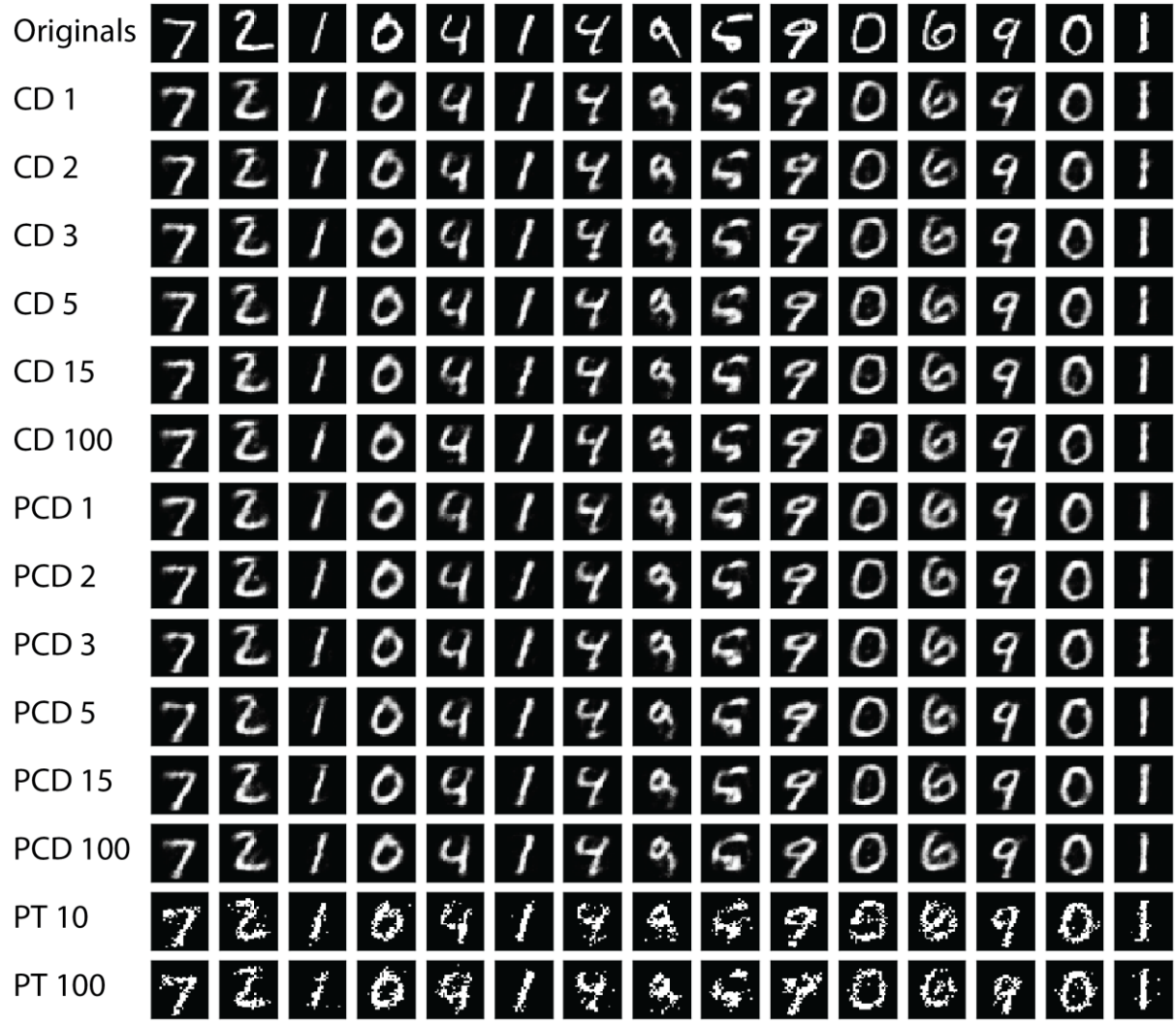



Рис. 2.5. Результати відновлення моделями зображень із тестового набору MNIST

Оскільки навчання ОМБ базується на обчисленні та застосуванні градієнтів функції помилки відтворення вхідного сигналу, наведеному у виразах (2.32) та (2.33), для покращення результатів навчання окремих ОМБ та ГМД вцілому має сенс розглянути розвинені алгоритми градієнтної оптимізації, що й було виконано у наступному розділі.

**Висновки по розділу**

У результаті проведення серії експериментів було визначено оптимальні сценарії пошуку гіперпараметрів навчання обмеженої машини Больцмана. Так, було визначено, що алгоритм PT підходить для навчання ОМБ лише у системах із значною кількістю обчислювальних одиниць, що можуть бути запущені паралельно.

# РОЗДІЛ 3. ПОКРАЩЕННЯ НАЛАШТУВАННЯ ГЛИБОКИХ МЕРЕЖ ДОВІРИ

## 3.1. Алгоритми оптимізації обмеженої машини Больцмана

Базовим методом оптимізації параметрів у глибокому навчанні є метод градієнтного спуску у поєднанні із методом зворотнього розповсюдження помилки. Однак ця комбінація має значні проблеми:

1) високий шанс потрапляння у локальний оптимум;
2) затухання градієнтів із зростанням глибини мережі;
3) «вибух» параметрів (вагів).

Через сталість параметру швидкості навчання $\lambda$ протягом процесу тренування моделі, неможливо змусити процес вийти із локального мінімуму, якщо $\lambda$ достатньо мала, та навпаки, неможливо зупинитись у прийнятному оптимумі, якщо $\lambda$ достатньо велика (рис. 3.1).

Проблем додає й той факт, що у складній моделі різні параметри можуть змінюватись по-різному: деяким потрібен менший крок навчання, у той же час, як для іншим – навпаки, більший. Така проблема особливо актуальна, коли вхідні дані не нормалізовані та не масштабовані (рис. 3.2).

Ці вади унеможливлюють застосування градієнтного методу у чистому вигляді для глибоких нейромережевих моделей.

Для уникнення проблем та покращення результатів цього методу проводяться дослідження щодо його розвитку, що вже призвело до винаходу цілої низки потужних модернізованих методів-оптимізаторів серед яких Стохастистичний Градієнтний Спуск із Моментом (SGDM), Прискорений градієнт Нестерова (NAG), Адаптивний градієнтний алгоритм (AdaGrad, AdaDelta), Розповсюдження Кореня Середнього Квадрату Градієнта (RMSProp), Оцінка Адаптивного Момента (AdaM), Адаптивний Момент Нестерова (NAdaM). Для реалізації перелічених вище алгоритмів необхідний розрахунок градієнту, компонентами якого є часткові похідні узагальненої помилки по ваговим коефіцієнтам, які розраховуються за формулами (2.33).

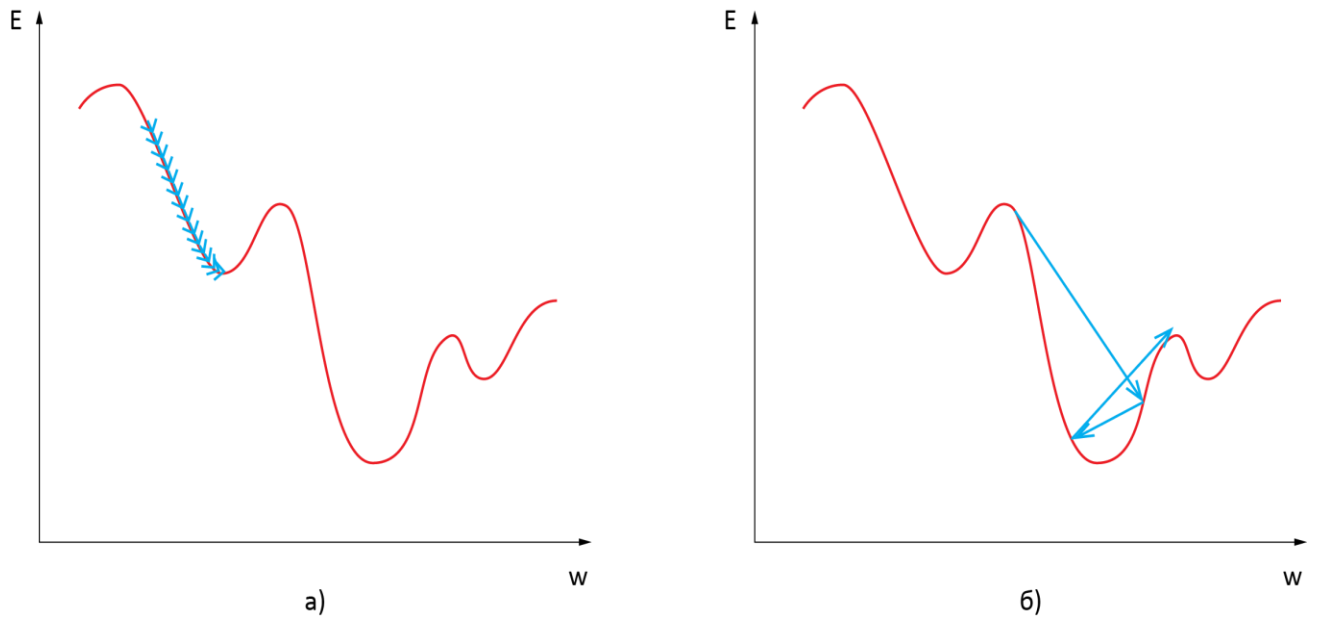



Рис. 3.1. Демонстрація проблем градієнтного метода оптимізації зі сталим кроком навчання: а) за умови занадто малого кроку навчання модель оптимізується дуже довго та дуже схильна залишитись у найближчому до точки старту локального оптимума функції втрат; б) за великого значення кроку навчання оптимізація не зупиняється у точці оптимума, навіть якщо він був досяжний

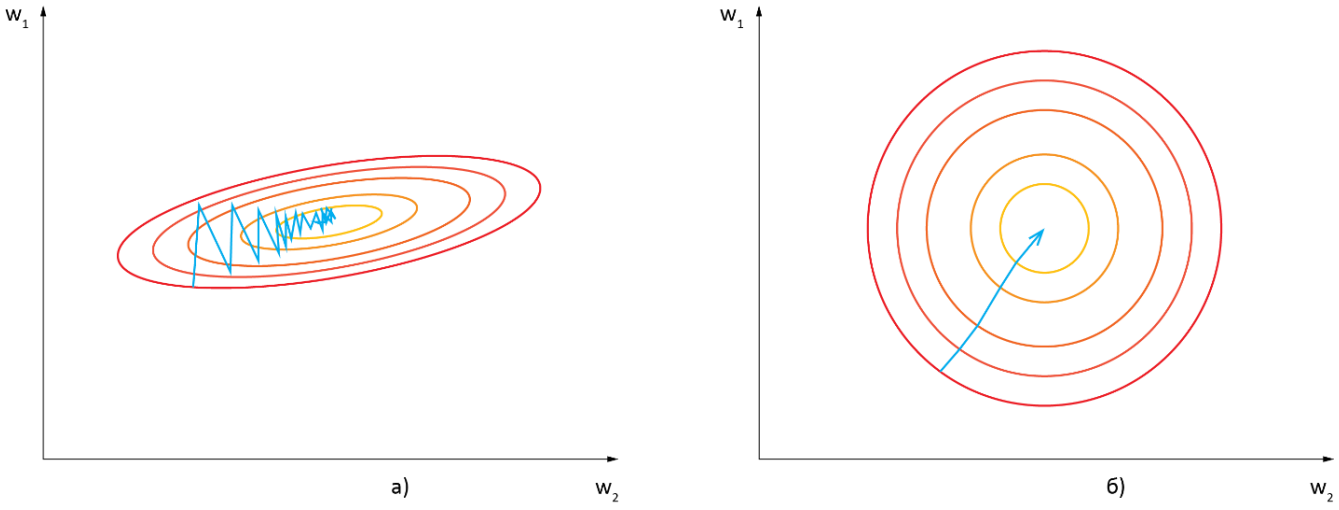



Рис. 3.2. Вплив нормалізації на якість оптимізації моделі за градієнтним методом: а) за різного масштабу сигналів, що входять у нейрон по зв'язках $w_1$ та $w_2$ відбувається «блукання» градієнта, що значно погіршує сходимість; б) після нормалізації входів функція втрат сходиться значно краще, завдяки рівномірному впливу вхідних сигналів

Розглянемо кожен метод оптимізації окремо.

### 3.1.1. Стохастичний Градієнтний Спуск із Моментом

Стохастичний Градієнтний Спуск із Моментом (Stochastic gradient descent with momentum, SGDM) є найближчим розвитком градієнтного метода, проте показує значно кращі результати оптимізації завдяки запровадженню двох еврістик:

1) розбиття тренувальних даних на рівнорозмірні підмножини даних (мінібатчі) та випадкове перемішування їх на кожній тренувалній епосі;
2) застосування Момента (Momentum) під час визначення величини оновлення параметрів на основі їх попередніх значень.

Ідея стохастичності цього метода полягає у оновленні параметрів мережі після проходження сигналу кожного зразка даних, взятого випадковим чином із загального тренувального датасета [17]. Таке навчання має назву поточного (on-line) тренування, проте має суттєвий недолік у вигляді значних витрат часу у випадку великих розмірів тренувального датасета для проходження сигналу у обох напрямах та обчислення градієнтів та виконання операцій оновлення параметрів для кожного зразка окремо.

$$\theta^t = \theta^{t-1} - \lambda \nabla E_k(\theta) \tag{3.1}$$

де, $k$ – окремий зразок даних, тут і далі $\nabla E$ – градієнт функції втрат, обчислений за виразом (2.33).

Для уникнення цих недоліків без втрати корисних властивостей алгоритма використовують розбиття датасета на групи зразків (міні-пакети, minibatches), які перед цим були випадковим чином перемішані [18].

У такому випадку $\nabla E_k$ – $k$-й мінібатч розміру $1 \ll m \ll M$ ($M$ – загальний обсяг датасета) якого обирається відносно великим, та зазвичай регулюється розмірами доступної пам’яті обчислювального пристрою, наприклад графічної карти тощо.

Такий підхід гарно паралелізується, оскільки кожен мінібатч можна обробити окремо, та для тренування використовувати розподілені обчислювальні системи.

Друга складова SGDM – Момент – додатковий параметр функції обчислення оновлених значень вагів мережі, який враховує попереднє значення зміни вагів [20].

$$\Delta\theta^t = v^t = \mu\Delta\theta^{t-1} - \lambda\nabla E(\theta), \tag{3.2}$$

$$\theta^t = \theta^{t-1} + v^t, \tag{3.3}$$

або

$$\theta^t = \theta^{t-1} - \lambda\nabla E(\theta) + \alpha\Delta\theta^{t-1}, \tag{3.4}$$

де $\mu$ – коефіцієнт момента, а $\Delta\theta^{t-1}$ попереднє значення оновлення параметрів мережі, $v^t$ – можна інтерпретувати як швидкість руху точки параметру по поверхні цільової функції (рис. 3.3).


Рис. 3.3. Схематичне зображення оптимізаційного пошуку по поверхні цільової функції а) без використання моменту та б) із використанням моменту

Назва «момент» взята як аналогія «момента інерції» з механіки: тензор параметрів $w$ мандрує ландшафтом цільової функції, отримуючи прискорення від градієнта цієї функції («сила»). На відміну від простого градієнтного спуску, це дозволяє продовжити рух із деякою інерцією у тому ж напрямі, уникаючи осциляції.

Момент вже кілька десятиліть успішно застосовується дослідниками для тренування штучних нейронних мереж.

### 3.1.2. Прискорений градієнт Нестерова

Прискорений градієнт Нестерова (Nesterov accelerated gradient, NAG) є розвитком попереднього метода. Винайдений ще у 1983 Нестеровим [21], є сьогодні досить популярним у машинному навчанні. SGDM спершу обчислює градієнт у поточній точці та потім виконує стрибок у напрямі оновленого

закумульованого градієнта. Натомість NAG спершу «стрибає» у напрямі попереднього стрибка із швидкістю, закумульованою на попередньому кроці, а лише потім обчислює нові градієнти. Іншими словами NAG імплементує стратегію «стрибок→корекція» (рис. 3.4).

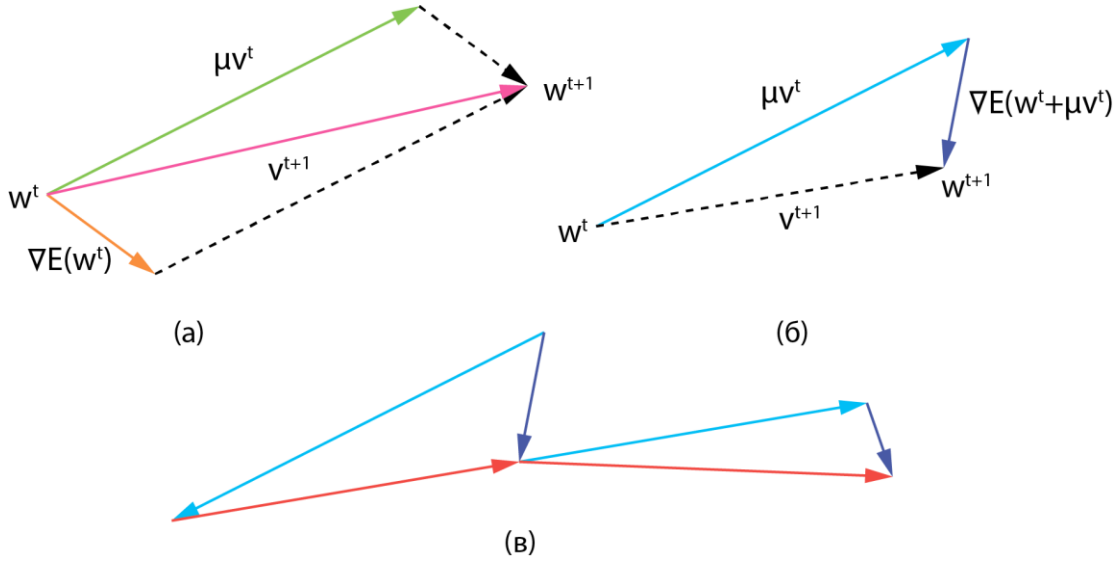



Рис. 3.4. Порівняння шляхів обчислення та оновлення значень вагів мережі для методів SGDM (а) та NAG (б), а також два кроки роботи NAG (в), де червона лінія демонструє акумульований градієнт

Формально це має наступний вигляд:

$$v^t = \mu v^{t-1} + \lambda \nabla E(\theta - \mu v^{t-1}), \tag{3.5}$$

$$\theta^t = \theta^{t-1} - v^t \tag{3.6}$$

Такий метод обчислення момента дозволяє уникати небажано високої швидкості зміни параметрів, що може призвести до вибухаючих градієнтів. Він наче передказує зміну параметрів, і якщо переміщення було завеликим та веде до гіршого стану цільової функції, він обчислює градієнт по параметру із урахуванням здійсненого переміщення. Також NAG ще краще бореться із зайвими осциляціями навколо точки оптимуму, він доведено краще (порівняно із ніж SGDM) працює на великій низці задач, особливо із рекурентними нейронними мережами [22]. Обидва методи працюють задовільно при відносно великому значеннях $\mu$ та $\lambda$, найчастіше у межах $\mu \in [0.9, 1)$ та $\lambda \in [0.01, 0.001]$ [23].

### 3.1.3. Адаптивний градієнтний алгоритм

Інакше Алгоритм адаптивних градієнтів (Adaptive gradient algorithm, AdaGrad) [24] – розвиток алгоритму SGDM, що має на меті управління таким параметром оптимізації як швидкість навчання (learning rate) $\lambda$. У всіх попередніх методах він є незмінним, та застосовується для усіх параметрів мережі на усіх тренувальних епохах. Це стає проблемою для глибоких та мереж із великою шириною шарів, коли кількість параметрів сягає мільйонів. Перша причина полягає у тому, що серед них є такі що відповідають більш та менш чутливим ознакам, тобто змінюються менше або більше, частіше чи рідше. Друга ж причина проблеми полягає у глибині мережі та затуханні градієнтів у шарах віддалених від її виходу, а отже такі градієнти необхідно посилювати достатньо для корисного оновлення параметрів глибоких шарів.

AdaGrad адаптивно змінює $\lambda$ для кожного параметра окремо, зменшуючи його для параметрів, що змінюються дуже часто, та збільшуючи для тих, що змінюються рідше. Тобто він намагається вирішити одразу дві проблеми градієнтних методів: затухання та вибухання градієнтів, причому індивідуально для кожного параметра.

$$\theta^{t+1} = \theta^t - \frac{\lambda}{\sqrt{\varepsilon + G^t}} \cdot g^t, \tag{3.7}$$

де $\lambda$ – початкова швидкість навчання, $\varepsilon$ – дуже мала величина, що запобігає діленню на нуль при $G^t = 0$, зазвичай $\varepsilon = 1.0e - 10$,

$$g^t = \nabla E(\theta^t), \tag{3.8}$$

$$G^t = \sum_{\tau=1}^{t} g^\tau g^{\tau\top}, \tag{3.9}$$

$g^t$ – градієнт цільової функції по параметру, а $G^t$ – кумулятивний градієнт по цьому параметру за усі епохи навчання, який є діагональною матрицею квадратів $g^\tau$.

Таким чином AdaGrad накопичує інформацію про зміну градієнтів по параметру та адаптує $\lambda$ згідно із цією інформацією.

Цей метод показує значно кращі результати на дуже глибоких моделях із сотнями мільйонів параметрів, ніж попередні, та, що дуже важливо, навіть для

неопуклих функцій, які часто зустрічаються при застосуванні нейромережевих моделей.

### 3.1.4. Розповсюдження Кореня Середнього Квадрату Градієнта

Розповсюдження Кореня Середнього Квадрату Градієнта (Root Mean Square Propagation, RMSProp) – оптимізаційний метод винайдений Хінтоном, з метою вирішення проблем, попередньо перелічених методів, включно із AdaGrad. Його робота, як і в AdaGrad полягає у адаптивній зміні швидкості навчання $\lambda$, проте він робить це у інший спосіб [19][25].

Цей метод базується на іншому, що має назву RProp, який полягає у «стимулюванні» градієнтів, що змінюються у одному напрямку кілька епох поспіль, та «пригніченні» таких, що часто змінюють знак. Оскільки різні градієнти змінюються по-різному, тож і крок навчання доречно використовувати індивідуально.

RProp працює лише із повним датасетом, без поділу на мінібатча, та використовує лише знак градієнта, а отже усі параметри тренуються із однаковою амплітудою градієнта, що дозволяє уникати плато для дуже малих значень градієнта. Якщо, наприклад, на двох останніх кроках навчання знак градієнта по параметру $w_i$ не змінювався, то для нього крок навчання $\lambda$ збільшується із коефіцієнтом $k = 1.2$, інакше зменшується із $k = 0.5$. Проте, оскільки оцінка зміни знаку по параметру відбувається усередненням по цілому датасету, використовувати RProp із мінібатчами немає сенсу, оскільки знак градієнта по параметру може змінюватись від мінібатча до мінібатча.

Для застосування цієї проблеми було розроблено RMSProp, у якому градієнти залишаються такими ж, як і у класичному градієнтному методі, а швидкість або крок навчання регулюється акумульованою швідкістю зміни градієнта на попередніх етапах.

Нехай

$$d\theta = \nabla E(\theta), \tag{3.10}$$

тоді

$$v_{d\theta} = \beta v_{d\theta} + (1-\beta) d\theta^2, \tag{3.11}$$

$$\theta = \theta - d\theta \frac{\lambda}{\sqrt{v_{d\theta} + \varepsilon}}, \tag{3.12}$$

де $\varepsilon$ – страховка від ділення на нуль при $v_{d\theta} = 0$.

Отже RMSProp допомогає уникати осциляцій градієнта по параметру та прискоряти пошук оптимуму, враховуючи, що він дозволяє використовувати достатньо великі значення кроку навчання $\lambda$.

### 3.1.5. Оцінка Адаптивного Момента

Оптимізаційний метод, що поєднує підходи методів AdaGrad та RMSProp. Він також проводить зміну кроків навчання індивідуально для параметрів, зважаючи на «історію» зміни градієнтів на них [26].

Нехай

$$v = \beta_1 v + (1-\beta_1) d\theta, \tag{3.13}$$

$$s = \beta_2 s + (1-\beta_2) d\theta^2, \tag{3.14}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_1^t}, \tag{3.15}$$

$$\widehat{s_t} = \frac{s_t}{1-\beta_2^t}, \tag{3.16}$$

де $v$ та $s$ називають першою та другою оцінками моменту, $\widehat{v_t}$ та $\widehat{s_t}$ – скореговані відносно зміщення оцінки моменту.

Тоді зміна параметрів відбувається наступним чином

$$\theta = \theta - \lambda \frac{\widehat{v_t}}{\sqrt{\widehat{s_t} + \varepsilon}} \tag{3.17}$$

Для багатьох особливо глибоких нейронних мереж цей алгоритм показує результати кращі, за інші оптимізатори, хоча для неглибоких мереж він не дає значних переваг порівняно із AdaGrad чи NAG.

Його переваги:

- Реальний крок навчання на кожній епосі є приближено пов’язаним із заданим кроком;

- Крок оновлення прямо не залежить від поточної величини градієнта, що дозволяє уникати плато, які є проблемою для SGD, який у них швидко застрягає;
- AdaM був розроблений для поєднання переваг AdaGrad, що добре працює із розрідженими градієнтами, та RMSProp, що добре працює на малих розмірах мінібатча. Також цей метод дуже легко поєднується із NAG (NAdaM) та SGDM.

Проте він має й недоліки. Недивлячись на дуже швидку сходимість [25], експериментально було виявлено, що AdaM іноді сходиться до гірших оптимумів, ніж SGDM чи NAG. Зокрема на датасетах сімейства CIFAR. Було проведено дослідження, які показали можливість заміни AdaM після певної кількості тренувальних епох на SGDM, який досягав кращого оптимума [27].

## 3.2. Вплив виду оптимізатора на навчання обмеженої машини Больцмана

Зважаючи на досягнення у розвитку оптимізаційних методів постає питання застосування їх для покращення навчання ОМБ, порівняно із звичайним градієнтим методом. Хоча деякі недоліки класичного градієнтного метода із зворотнім розповсюдженням помилки нівелюються стохастичною природою ОМБ, застосування моменту, оцінка градієнту на основі історії тренування, керування кроком навчання у процесі тренування заслуговують дослідження стосовно використання для тренування ОМБ.

Для цього проведено низку експериментів із використанням еталонного датасета CIFAR-10 [31] на експериментальній моделі ОМБ із параметрами, наведеними у табл. 3.1. Структура моделі для оцінки впливу виду оптимізатора на навчання ОМБ співпадає із структурою експериментальної моделі для попередніх експериментів у розділі 2 (рис. 2.4). Процес валідації результатів експериментів також співпадає із попереднім, а отже проведено оцінювання генеративної та класифікаційної здатності ОМБ за відповідними критеріями.

Таблиця. 3.1. Параметри експериментальної моделі ОМБ та гіперпараметри оптимізації для визначення впливу виду оптимізатора на процес тренування

| Параметр | Значення |
|---|---|
| Розмір видимого шару | 3072 |
| Розмір прихованого шару | 1000 |
| Розмір шару класифікатора | 10 |
| Розмір тренувального пакету | 100 |
| Кількість навчальних епох | 50 |
| Кількість навчальних епох контрольного класифікатора | 30 |
| Крок навчання | 0.001 |
| Алгоритм ОМБ | CD-1 |
| Розмір тренувального набору даних | 50000 |
| Розмір тестового набору даних | 10000 |

### 3.2.1. SGD та SGDM

Для алгоритму SGDM було проведено 4 експерименти із значеннями моменту $\mu = \{0.1, 0.5, 0.9, 0.99\}$. Результати експерименту подано у табл. 3.2 та у додатку Д.

Таблиця. 3.2. Результати порівняльних експериментів для різних значень моменту у алгоритмі SGDM

| Момент | Похибка генерації | | Точність класифікації | |
|---|---|---|---|---|
| | 1 епоха | остання епоха | 1 епоха | остання епоха |
| 0.0 (SGD) | 0.050145 | 0.027080 | 0.3432 | 0.4405 |
| 0.1 | 0.049340 | 0.026630 | 0.3477 | 0.4453 |
| 0.5 | 0.045789 | 0.024079 | 0.3397 | 0.4524 |
| 0.9 | 0.036009 | 0.015268 | 0.3473 | 0.4430 |
| 0.99 | 0.033236 | 0.013347 | 0.3421 | 0.4647 |

Результати експериментів показують принципову перевагу алгоритму SGDM над SGD, при чому із значенням моменту 0.99 оптимізатор відшукує найкращий оптимум функції втрат, до того ж значно швидше за інші варіанти.

### 3.2.2. SGD та NAG

Для алгоритму NAG також було проведено 4 експерименти із значеннями моменту $\mu = \{0.1,0.5,0.9,0.99\}$. Результати експерименту подано у табл. 3.3 та у додатку Д. Вони дуже схожі на результати порівняння алгоритмів SGDM над SGD. Алгоритм NAG показує принципово кращі результати порівняно із SGD, та трохи кращі відносно SGDM при тих же самих значеннях гіперпараметрів моменту та кроку навчання.

Таблиця. 3.3. Результати порівняльних експериментів для різних значень моменту у алгоритмах NAG та SGD

| Момент | Алгоритм | Похибка генерації | | Точність класифікації | |
|---|---|---|---|---|---|
| | | 1 епоха | остання епоха | 1 епоха | остання епоха |
| 0.0 | SGD | 0.050145 | 0.027080 | 0.3432 | 0.4405 |
| 0.1 | NAG | 0.049338 | 0.026631 | 0.3375 | 0.4423 |
| 0.5 | NAG | 0.045778 | 0.024078 | 0.3482 | 0.4523 |
| 0.9 | NAG | 0.035837 | 0.015240 | 0.3438 | 0.4430 |
| 0.99 | NAG | 0.033236 | 0.013347 | 0.3300 | 0.4678 |

### 3.2.3. SGD та RMSProp

Для алгоритму RMSProp також було проведено 4 експерименти із значеннями коефіцієнта $\beta = \{0.1, 0.5, 0.9, 0.99\}$. Результати експерименту подано у табл. 3.4 та у додатку Д.

У результаті експериментів можна зазначити принципову перевагу алгоритма RMSProp над SGD, проте також на графіку помітна волатильність або блукання RMSProp навколо оптимумів, що свідчить про занадто великий крок навчання для цього алгоритма із цією моделлю та набором даних.

Таблиця. 3.4. Результати порівняльних експериментів для різних значень моменту у алгоритмах RMSProp та SGD

| Момент | Алгоритм | Похибка генерації | | Точність класифікації | |
|---|---|---|---|---|---|
| | | 1 епоха | остання епоха | 1 епоха | остання епоха |
| 0.0 | SGD | 0.050145 | 0.027080 | 0.3432 | 0.4405 |
| 0.1 | RMSProp | 0.031899 | 0.018290 | 0.3486 | 0.4593 |
| 0.5 | RMSProp | 0.030798 | 0.016634 | 0.3517 | 0.4635 |
| 0.9 | RMSProp | 0.030436 | 0.015698 | 0.3585 | 0.4776 |
| 0.99 | RMSProp | 0.029422 | 0.015679 | 0.3522 | 0.4744 |

### 3.2.4. SGD та AdaGrad

Було проведено два експерименти із однаковими початковими налаштуваннями спільних параметрів. Результати експерименту подано у табл. 3.5 та у додатку Д.

Таблиця 3.5. Результати порівняльних експериментів для алгоритмів SGD та AdaGrad із однаковими гіперпараметрами

| Алгоритм | Похибка генерації | | Точність класифікації | |
|---|---|---|---|---|
| | 1 епоха | остання епоха | 1 епоха | остання епоха |
| SGD | 0.050145 | 0.027080 | 0.3432 | 0.4405 |
| AdaGrad | 0.037363 | 0.026286 | 0.3528 | 0.4534 |

У результаті експериментів помітно, що алгоритм AdaGrad не дає значної якісної переваги відносно алгоритма SGD при застосуванні для тренування ОМБ на даному датасеті.

### 3.2.5. SGD та AdaM

Було проведено два експерименти із однаковими початковими налаштуваннями спільних параметрів, та із рекомендованими параметрами $\beta_1 = 0.9$ та $\beta_2 = 0.99$ алгоритма AdaM [26]. Результати експерименту подано у табл. 3.6 та у додатку Д. Вони свідчать про принципову перевагу застосування алгоритму AdaM над SGD навіть без додаткового налаштування параметрів AdaM.

Таблиця. 3.6. Результати порівняльних експериментів для алгоритмів SGD та AdaM із схожими налаштуваннями

| Алгоритм | Похибка генерації | | Точність класифікації | |
|---|---|---|---|---|
| | 1 епоха | остання епоха | 1 епоха | остання епоха |
| SGD | 0.050145 | 0.027080 | 0.3432 | 0.4405 |
| AdaM | 0.026042 | 0.014054 | 0.3438 | 0.4503 |

Табл. 3.7 та додаток Д містять зведені результати порівняння впливу вибору оптимізатора на сходимість обмеженої машини Больцмана, при спільних налаштуваннях кроку навчання $\lambda = 0.001$, кількості тренувальних епох рівній 50 та алгоритмі навчання ОМБ CD-1. Для їх формування було взято найкращі за ефективністю гіперпараметри для кожного алгоритма оптимізації із попередніх експериментів.

Таблиця. 3.7. Зведені результати порівняльних експериментів для різних алгоритмів оптимізації ОМБ

| Момент | Алгоритм | Похибка генерації | | Точність класифікації | |
|---|---|---|---|---|---|
| | | 1 епоха | остання епоха | 1 епоха | остання епоха |
| 0.0 | SGD | 0.050145 | 0.027080 | 0.3432 | 0.4405 |
| 0.99 | SGDM | 0.033237 | 0.013348 | 0.3421 | 0.4647 |
| 0.99 | NAG | 0.030135 | 0.013168 | 0.3300 | 0.4678 |
| 0.0 | AdaGrad | 0.037363 | 0.026286 | 0.3528 | 0.4534 |
| 0.99 | RMSProp | 0.029422 | 0.015679 | 0.3522 | 0.4744 |
| 0.0 | AdaM | 0.026042 | 0.014055 | 0.3438 | 0.4503 |

Найкращий результат оптимізації було досягнуто алгоритмами NAG та SGDM із значенням моменту $\mu = 0.99$. Алгоритм AdaM також показав близький до кращого результат із рекомендованими розробниками налаштуваннями за

замовчуванням $\beta_1 = 0.9$ та $\beta_2 = 0.99$. Алгоритм RMSProp демонструє блукання навколо оптимумів цільової функції, що може свідчити про необхідність пошуку більш прийнятних значень кроку навчання та моменту, проте також дає значно кращі результати, порівняно із стандартним градієнтним спуском. Не зважаючи на позитивні оцінки дослідників алгоритму AdaGrad для глибоких нейронних мереж, у випадку застосування для ОМБ він не показує вагомих переваг перед алгоритмом SGD без моменту.

На рис. 3.5 показана генеративна здатність ОМБ після 50 навчальних епох із застосуванням різних оптимізаторів. Зразки даних взяті з тестового набору даних CIFAR-10, який не приймав участь у навчанні моделей.

Оскільки вибір алгоритму оптимізації для тренування ОМБ очевидно має принципове значення, цей процес також має бути включений у решітку пошуку гіперпараметрів навчання моделі.

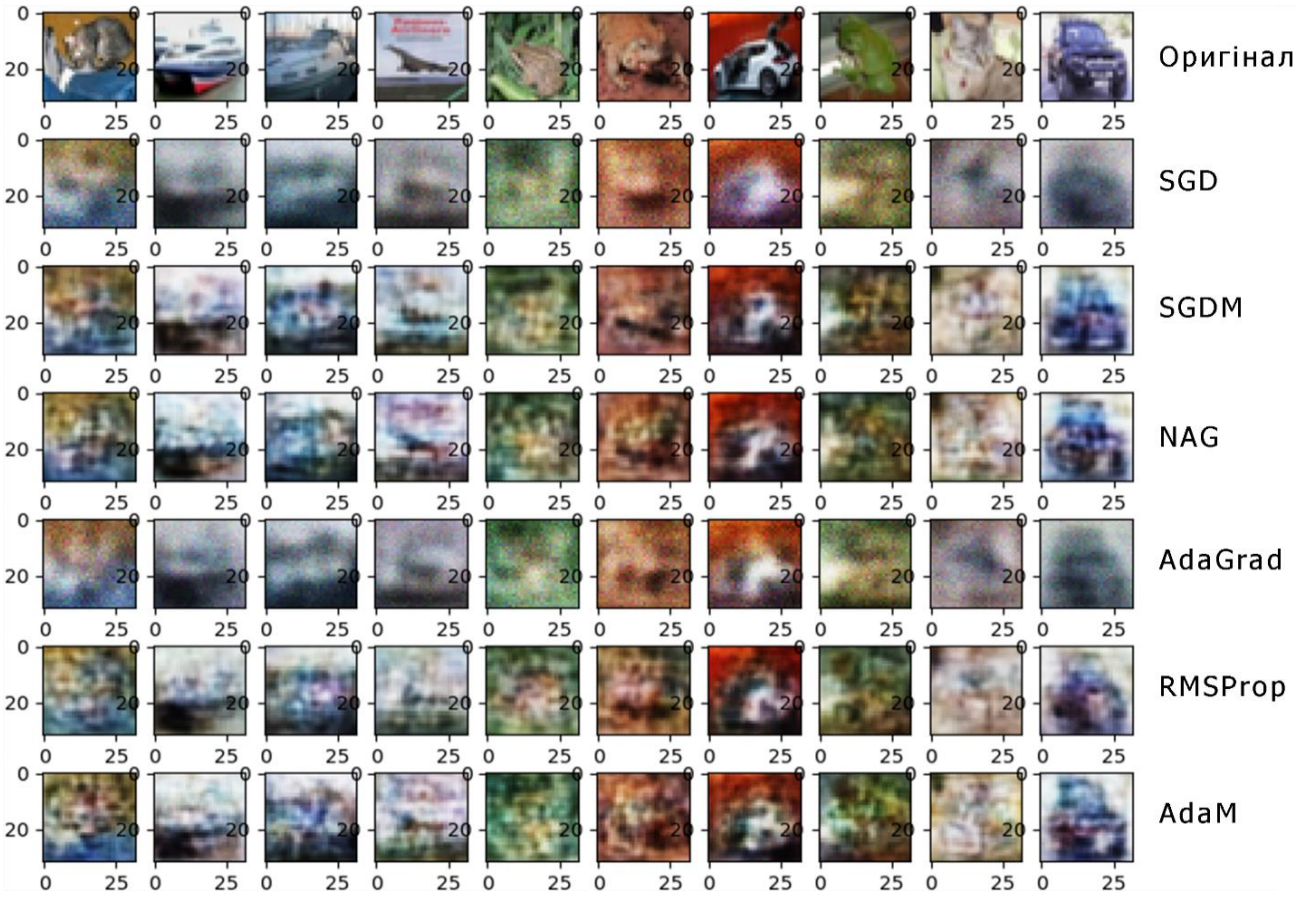



Рис. 3.5. Порівняння генеративної здатності ОМБ навченої із різними алгоритмами оптимізації

**Висновки по розділу**

Результатом дослідження впливу алгоритмів оптимізації навчання ОМБ став розвиток первинної стратегії структурно-параметричного синтезу ГМД.

У оновленій стратегії у процесі найкращого налаштування кожної ОМБ у каскаді необхідно визначати не лише оптимальний базовий алгоритм навчання моделі (CD, PCD, PT), а й наступним кроком визначати найкращий оптимізаційний алгоритм (SGDM, NAG, AdaGrad, AdaM, RMSProp). Таким чином кінцева стратегія структурно-параметричного синтезу ГМД приймає наступний вигляд:

1) Ініціалізація ГМД;
2) Додавання ОМБ до існуючої структури. Якщо це перша ланка каскаду, то вхідні дані формуються безпосередньо із тренувального датасета, інакше, вхідні дані формуються проходженням даних тренувального датасета крізь попередні ОМБ у каскаді;
3) Тренування поточної ОМБ:
    3.1) Пошук кращого базового алгоритму навчання ОМБ;
    3.2) Пошук кращого алгоритму та набору гіперпараметрів оптимізації ОМБ;
    3.3) Остаточне навчання ОМБ із найкращими знайденими гіперпараметрами;
4) Додавання шару класифікатора та дискримінантне донавчання ГМД;
5) Перевірка усієї ГМД на якість за критерієм мінімізації похибки узагальнення шляхом валідації генеративної та класифікаційної здатності мережі на тестовому наборі даних;
6) Якщо похибка узагальнення перевищує цільову, шар класифікатора видаляється, існуючі параметри мережі фіксуються та повторюються кроки 2–6.

Після досягнення необхідної якості ГМД процес структурно-параметричного синтезу вважається закінченим, а модель – побудованою та навченою.

Остаточний алгоритм структурно-параметричного синтезу глибокої мережі довіри наведено у додатку А.

# РОЗДІЛ 4. РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Система оптимального налаштування глибоких мереж довіри (СОНГМД) створена як компонент для програмної системи робототехнічного комплексу. Схему такої системи наведено у додатку Б.

Зі схеми видно, що СОНГМД взаємодіє із модулем розпізнавання та класифікації вхідного зображення, та відповідно, призначений для його налаштування.

## 4.1. Функціональні вимоги до програмного забезпечення

Система оптимального налаштування глибокої мережі довіри має виконувати наступні задачі:

1) Завантаження набору даних;
2) Приведення набору даних до необхідного формату (векторизація, аугментація, нормалізація тощо);
3) Розбиття вхідних даних на тренувальний та тестовий набори;
   3.1) Додаткове розбиття набору даних у випадку наявності нерозміченої частини даних;
4) Пошук оптимальних гіперпараметрів навчання моделей;
5) Побудова та тренування цільових нейромережевих моделей: ГМД, БШП, ОМБ;
6) Збереження та експорт моделей у придатному до виробничого використання вигляді.

## 4.2. Архітектура системи оптимального налаштування глибоких мереж довіри

Система оптимального налаштування глибоких мереж довіри складається з наступних компонентів:

- Диспетчер;
- Попередньої обробки тренувальних даних;

- Структурної побудови нейромережевих моделей: ГМД, ОМБ, БШП;
- Навчання ОМБ, ГМД, БШП;
- Експорту та імпорту моделей та результатів тренування;
- Візуалізації, призначений для демонстрації та оцінки результатів налаштування моделей користувачем.

Схема системи розміщена у додатку В.

Компонент «Диспетчер» виконує інтегральну функцію та підтримує архітектурну незалежність інших компонентів. Він інкапсулює логіку управління життєвим циклом системи, здійснює виклики компонентів із передачею їм вхідних даних та отримання результатів їх роботи.

Компонент попередньої обробки тренувальних даних здійснює наступні дії:

- розбиття даних на датасети (тренувальний, валідаційний, тестовий) залежно від конфігурації, заданої диспетчером;
- масштабування та нормалізацію даних;
- векторизацію зразків.

Компонент структурної побудови нейромережевих моделей отримує на вхід початкову конфігурацію та на її основі створює програмні об’єкти заданих моделей:

- Обмежена машина Больцмана – під час ініціалізації створюються параметри самої моделі, та задаються первинні налаштування для тренування моделі.
- Глибока мережа довіри – якщо від диспетчера були отримані лише декларативні параметри мережі, компонент створює каскад ОМБ на основі цих аргументів, та виконує первинну ініціалізацію параметрів усієї мережі. Якщо диспетчер передав у компонент набір готових об’єктів ОМБ, компонент створює ГМД із переданого набору, а також додаткових параметрів ініціалізації та навчання ГМД. Останній варіант створення ГМД – перетворення його із багатошарового персептрона у каскад ОМБ для подальшого тренування згідно із заданими аргументами.

- Багатошаровий персептрон – також може бути створений різних способом при різних аргументах. Наприклад, якщо диспетчер передав готовий об’єкт ГМД, компонент конвертує ГМД у БШП, передаючи таким чином результати переднавчання цільовій моделі. Якщо ж від диспетчера отримано лише конфігурацію цільової моделі, згідно із нею відбувається повна побудова та ініціалізація об’єкта БШП.

Результатом роботи компонента є готовий об’єкт запрошеної моделі, що передається до диспетчера.

Компонент навчання глибокої мережі довіри. Він отримує від диспетчера об’єкт моделі ГМД, виконує його декомпозицію на об’єкти ОМБ та передає їх для навчання у компонент конкурентного навчання ОМБ. Після навчання кожної ланки каскаду компонент пропускає тренувальні дані крізь неї та конвертує їх у тренувальні дані для наступної ланки.

Компонент навчання цільової моделі – Багатошарового персептрона – отримує об’єкт моделі БП та додаткові аргументи залежно від яких виконує наступні дії:

1) Визначає, чи модель була сформована на базі попередньо навченої ГМД;
2) Складає карту експериментів, що містить усі можливі комбінації навчальних параметрів;
3) Виконує незалежно усі експерименти;
4) Порівнює їх результати та визначає найкращий за критерієм досягнення найкращого оптимуму функції втрат шляхом крос-валідації.

Після завершення тренування готовий об’єкт БП повертається до диспетчера.

Компонент конкурентного навчання ОМБ отримує об’єкт ОМБ від диспетчера. Якщо від диспетчера було передано бажану конфігурацію навчання ОМБ, то подальше навчання відбувається за нею. Інакше, компонент використовує власні базові налаштування конкурентного навчання ОМБ, яке складається за наступним алгоритмом:

1) Складається карта експерименту, що вміщує в себе усі можливі комбінації навчальних гіперпараметрів;

2) Створюється достатня кількість копій базового об’єкта ОМБ для виконання ізольованих експериментів із кожною комбінацією гіперпараметрів.
3) Виконується проведення експериментів незалежно та паралельно (зважаючи на доступність обчислювальних ресурсів).

Після закінчення усіх експериментів відбувається порівняння їх результатів та вибір найкращого за певним критерієм. Якщо інше не задано диспетчером, критерієм оптимальності виступає мінімум функції втрат за задану кількість тренувальних епох.

ОМБ, що була отримана у результаті найкращого експерименту повертається диспетчеру.

Компонент візуалізації призначений для надання користувачеві візуальної інформації про результати навчання моделей. Він містить набір утиліт для порівняння результатів експериментів із різними налаштуваннями та допомагає зробити висновки про адекватність отриманих результатів.

Компонент експорту / імпорту моделей та результатів призначений для збереження, серіалізації, десеріалізації моделей та результатів експериментів, а також для відсилання їх сторонньому адресату чи отримання із сторонного ж джерела.

Система забезпечує програмний інтерфейс для отримання команд та повернення результатів, та може виступати підсистемою іншої системи, забезпечуючи створення або покращення заданих нейромережевих об’єктів.

### 4.3. Конкурентне навчання моделей

Для визначення оптимального набору тренувальних гіперпараметрів застосовується пошук по решітці із повним перебором їх комбінацій. Такий пошук супроводжується вимірюванням продуктивності за критерієм досягнення оптимуму функції втрат та мінімум витрат часу. Якість навчання моделі для кожної комбінації перевіряється шляхом кросвалідації або валідації на стабільній валідаційній виборці, якщо така представлена для тренування.

Простір пошуку гіперпараметрів може бути представлений як дискретними так і не перервними множинами, а отже система повинна забезпечувати дискретизацію неперервних множин для забезпечення ефективного пошуку із припустими витратами обчислювальних ресурсів.

Оскільки такий пошук можливо виконувати паралельно, доречно застосовувати обчислювальні системи, що дозволяють виконувати паралельне тренування та валідацію моделей із різними гіперпараметрами одночасно.

Для навчання обмеженої машини Больцмана система має наступні гіпараметри, що утворюють простір для організації решітки пошуку:

- Алгоритми навчання ОМБ $Algs = \{\mathrm{CD}k, \mathrm{PCD}k\}$;
- Глибина Марківського ланцюга ОМБ $k = \{1, 5, 15\}$;
- Алгоритми оптимізації: $Opts = \{\mathrm{SGDM}, \mathrm{NAG}, \mathrm{Adam}, \mathrm{AdaGrad}, \mathrm{RMSProp}\}$;
- Параметри налаштування алгоритмів оптимізації:
    - Крок навчання $lr = \left\{10^{-i} \middle| 3 \leq i \leq 5, i \in \mathbb{N}\right\}$;
    - Момент $m = \{0.9, 0.99\}$.

Повний розмір решітки пошуку $gs_{RBM} = 180$ комбінацій для ОМБ.

Для остаточного навчання (fine tunning) ГМД набір гіперпараметрів представлений меншим переліком:

- Алгоритми оптимізації: $Opts = \{\mathrm{SGDM}, \mathrm{NAG}, \mathrm{Adam}, \mathrm{AdaGrad}, \mathrm{RMSProp}\}$;
- Параметри налаштування алгоритмів оптимізації:
    - Крок навчання $lr = \left\{10^{-i} \middle| 3 \leq i \leq 5, i \in \mathbb{N}\right\}$;
    - Момент $m = \{0.9, 0.99\}$.

Повний розмір решітки пошуку $gs_{DBN} = 30$ комбінацій для ГМД.

Для вибору оптимальної комбінації гіперпараметрів достатньо провести випробування із невеликою (до 50) кількістю тренувальних епох. Після чого найкращий варіант донавчають до точки зупинки:

- досягнення мінімального значення функції втрат;
- досягнення максимальної кількості тренувальних епох.

## 4.4. Інтерфейс, вхідні та вихідні дані

Програмний продукт орієнтовано на досвідченого користувача із навичками роботи із застосуванням менеджерів командного рядка. Для запуску програми потрібно виконати консольний виклик скрипта та передати йому словник аргументів, що містить бажану конфігурацію запуску програми.

Вхідні дані що можуть бути застосовані у програмному продукту розділяються на 3 типи:

- Словник аргументів, що містять параметри бажаної конфігурації роботи програми;
- Набір даних для створення та тренування цільової моделі;
- Модель, попередньо збережена у відповідному форматі (опціонально).

Словник аргументів для конфігурації програми наведено у табл. 4.1.

Параметр «search» включає (1) або виключає (0) режим автоматичного пошуку по решітці оптимальної конфігурації у просторі варіантів. При значенні 0 програма виконує побудову та навчання цільової моделі із налаштуваннями за замовчуванням, якщо словник вхідних параметрів не містить відповідних параметрів побудови та навчання моделі.

Таблиця 4.1. Словник аргументів із можливими значеннями для конфігурації запуску програми

| Ключ | Можливі значення | Обов’язково |
|---|---|---|
| action | {build, build_and_train} | так |
| target | {rbm, dbn, mlp} | так |
| dataset | рядок символів, що містить шлях до набору даних | так |
| hiddens | кортеж цілих чисел, розділених комою | ні |
| search | {0, 1} | ні |
| save_to | рядок символів, що містить шлях до цільової директорії для збереження результатів | ні |
| epochs | ціле число | ні |
| batch_size | ціле число | ні |
| lr | крок навчання | ні |
| opt | {sgdm, nag, rmsprop, adagrad, adam} | ні |
| help | {∅, 1} | ні |

Параметр «save_to» може бути застосований, якщо користувач має намір зберегти результати роботи системи у директорію, що відрізняється від директорії за замовчуванням.

Параметри «hiddens», «epochs», «batch_size», «lr», «opt» є необов'язковими та призначені для перевантаження значень за замовчуванням параметрів функціонування системи, особливо у випадку відключення режиму автоматичного пошуку оптимального налаштування цільової моделі.

## 4.5. Інструменти програмної реалізації

Для програмної реалізації системи оптимального налаштування ГМД було використано наступні інструменти:

- Мова програмування та віртуальна машина Python, version 3.6;
- Фреймворк глибокого навчання PyTorch, version 1.3;
- Бібліотека обчислень на графічних адаптерах CUDA, version 10.1;
- Бібліотека тензорних обчислень NumPy, version 1.16;
- Бібліотека візуалізації Matplotlib, version 3.1;
- Фреймворк досліджень Jupyter.

## 4.6. Контрольні приклади

Для валідації роботи системи оптимального налаштування ГМД було виконано два контрольних приклади із застосуванням датасетів CIFAR-10 [31] та Fruits-360 [32].

Зважаючи на способи застосування ГМД для попереднього навчання БШП, описані у розділі 1, було прийнято рішення про проведення низки експериментів. Для експериментів із частково-розміченим набором даних його було отримано шляхом відкидання міток для частини набору, що імітувала нерозмічену частину.

Кожний контрольний приклад складається 4 експериментів:

1) Налаштування ГМД на повністю розміченому наборі даних, перетворення її у БШП із переднавчанням та остаточного налаштування;

2) Налаштування ГМД на частково-розміченому наборі даних, перетворення її у БШП із переднавчанням та остаточного налаштування;
3) Налаштування БШП на повністю розміченому наборі даних, без переднавчання за допомогою ГМД;
4) Налаштування БШП на частково-розміченому наборі даних, без переднавчання за допомогою ГМД.

Кожний експеримент було проведено із ітеративним нарощуванням глибини мережі ГМД до досягнення найкращого значення точності класифікації на тестовому наборі. Для економії обчислювальних витрат максимальну ширину прихованого шару було обмежено значенням $n = 1000$ нейронів. Цього виявилось достатньо для порівняльних експериментів, враховуючи можливість збільшувати ємність моделі у глибину, та достатність отримання відносних показників ефективності моделі порівняно із базовою моделлю БШП без переднавчання.

Обчислювальна система, що була задіяна для проведення контрольних експериментів представлена у табл. 4.2.

Таблиця 4.2. Конфігурація обчислювальної системи, задіяної для проведення контрольних експериментів

| | |
|---|---|
| Процесор | Intel Core i7-8750H, 6 cores |
| Оперативна пам’ять | 32 ГБ, DDR4 |
| Накопичувач інформації | SSD, 256 GB |
| Графічний адаптер | NVIDIA GeForce GeForce GTX 1060, 6 GB |

### 4.6.1. CIFAR-10

Параметри набору даних CIFAR-10 наведено у табл. 4.3.

Таблиця 4.3. Параметри набору даних CIFAR-10

| | |
|---|---|
| Загальний обсяг | 60000 |
| Тренувальний набір | 50000 |
| Тестовий набір | 10000 |
| Зразок даних, розмір | 3x32x32 |
| Зразок даних, формат | 3 канальне зображення, RGB, JPG |

Імітаційний набір даних для проведення експериментів із повністю та частково розміченими наборами було отримано після розділу початкового набору даних. Його параметри наведено у табл. 4.4, а приклади зразків даних наведено на рис. 4.1.

Кожний експеримент було проведено із ітеративним нарощуванням глибини мережі ГМД до досягнення максимального значення точності класифікації, подальше ускладнення не має сенсу, оскільки веде до перетренованості моделі, що погіршує її якість.

Рис. 4.1. Зразки даних із набору CIFAR-10

Таблиця 4.4. Параметри імітаційного набору частково розмічених даних CIFAR-10

| | |
|---|---|
| Загальний обсяг | 60000 |
| Нерозмічений набір | 54000 |
| Розмічений набір | 6000 |
| Тренувальний набір для навчання без вчителя (unsupervised learning) | 50000 |
| Тестовий набір для навчання без вчителя | 10000 |
| Тренувальний набір для навчання із вчителем (supervised learning) | 5000 |
| Тестовий набір для навчання із вчителем | 1000 |

Результати експериментів наведено у табл. 4.5

За результатами експериментів на частково розміченому наборі даних очевидна перевага застосування ГМД для переднавчання із застосуванням нерозміченої частини для навчання без вчителя, або екстракції латентних ознак із усього обсягу даних. Таке попереднє налаштування параметрів нейронної мережі дає можливість отримати більш якісний результат наступного раунду її навчання із вчителем на розміченій частині набору.

Таблиця 4.5. Результати порівняльних експериментів моделей БШП із переднавчанням із використанням ГМД та БШП без перднавчання на напіврозміченому наборі даних

| Модель | Кількість прихованих шарів | Точність класифікації |
|---|---|---|
| БШП із переднавчанням з ГМД | 1 | 0.4640 |
| | 2 | 0.4700 |
| | 3 | 0.4710 |
| | 4 | 0.4770 |
| | 5 | 0.4680 |
| БШП без переднавчання | 1 | 0.4050 |
| | 2 | 0.4050 |
| | 3 | 0.4130 |
| | 4 | 0.3900 |
| | 5 | 0.3820 |

На повністю розміченому наборі даних моделі досягли найкращої якості при заданих обмеженнях вже із одним прихованим шаром. Подальше поглиблення мережі не покращило досягнутих результатів для обох моделей, навчання було зупинено. Результати експериментів наведено у табл. 4.6.

У результаті проведених експериментів на наборі даних CIFAR-10 можна зробити висновок, що застосування ГМД для переднавчання класифікатора має сенс, оскільки для цього набору даних він досягає кращих результатів, порівняно із

ідентичною за структурою моделлю, що навчалася без застосування ГМД, особливо у випадку сильної обмеженості розміченої частини набору даних.

Таблиця 4.6. Результати порівняльних експериментів моделей БШП із переднавчанням із використанням ГМД та БШП без перднавчання на повністю розміченому наборі даних

| Модель | Кількість прихованих шарів | Точність класифікації |
|---|---|---|
| БШП із переднавчанням з ГМД | 1 | 0.5705 |
| | 2 | 0.5632 |
| БШП без переднавчання | 1 | 0.5516 |
| | 2 | 0.5461 |

### 4.6.2. FRUITS

Параметри набору даних оригінального FRUITS-360 наведено у табл. 4.7.

Таблиця 4.7. Параметри набору даних FRUITS-360

| | |
|---|---|
| Загальний обсяг | 82110 |
| Тренувальний набір | 61488 |
| Тестовий набір | 20622 |
| Зразок даних, розмір | 3x100x100 |
| Зразок даних, формат | 3 канальне зображення, RGB, JPG |

У результаті експериментів було виявлено, що обидві моделі швидко сходяться до високого результату класифікації із точністю 0.9517 для БШП із ГМД та 0.9518 для БШП без переднавчання вже із одним прихованому шарі, а отже жодна модель не демонструє перевагу. Це пояснюється високою якістю набору даних:

- зразки не місять шуму та мають білий фон;
- надлишковість – кожен клас має велику кількість зразків;
- збалансованість – усі класи представлені приблизно рівною кіль-кістю зразків.

Приклади оригінальних зразків даних подано на рис. 4.2.

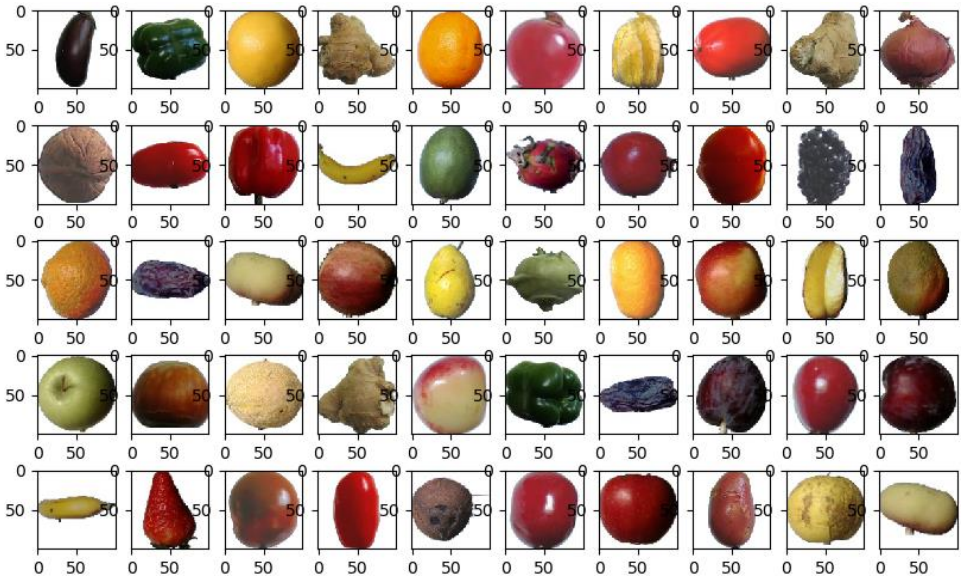


Рис. 4.2. Оригінальні зразки з набору даних, на яких обидві моделі показують однаково високу точність близько 95%

Для отримання більш виразних результатів експериментів набір даних було модифіковано із використанням наступних прийомів:

- пониження розмірності зразка до значення 3x10x10;
- додавання рівномірного білого шуму.

Приклади модифікованих зразків даних подано на рис. 4.3.

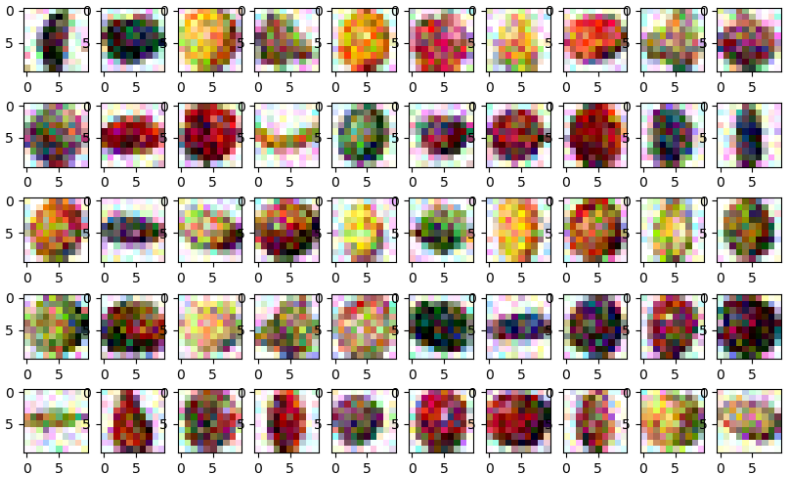


Рис. 4.3. Модифіковані зразки з набору даних, які дозволяють оцінити різницю ефективності моделей.

Імітаційний частково-розмічений набір, описаний у табл. 4.8.

Таблиця 4.8. Параметри імітаційного модифікованого набору частково-розмічених даних Fruits-360

| | |
|---|---|
| Загальний обсяг | 82110 |
| Нерозмічений набір | 73899 |
| Розмічений набір | 8211 |
| Тренувальний набір для навчання без вчителя (unsupervised learning) | 61488 |
| Тестовий набір для навчання без вчителя | 20622 |
| Тренувальний набір для навчання із вчителем (supervised learning) | 6149 |
| Тестовий набір для навчання із вчителем | 2062 |

Схожим чином із попередніми експериментами, результати експериментів на частково розміченому наборі даних демонструють перевагу застосування ГМД для переднавчання із застосуванням нерозміченої частини для навчання без вчителя, або екстракції латентних ознак із усього обсягу даних. Таке попереднє налаштування параметрів нейронної мережі дає можливість отримати більш якісний результат наступного раунду її навчання із вчителем на розміченій частині набору. До того ж, БШП без перенавчання демонструє найкращий результат на першому шарі, а збільшення глибини веде у даному випадку лише до перетренування моделі. ГМД ж робить цільову модель більш стійкою до перетренування, та показує покращення точності класифікації із ростом глибини до третього шару (табл. 4.9).

На повністю розміченому наборі даних БШП із переднавчання з ГМД досягла кращого результату порівняно із БШП без переднавчання вже із одним прихованим шаром. Найкращий результат було досягнуто із трьома прихованими шарами. Модель без переднавчанням показала свій найкращий результат, маючи два

прихованих шари, та подальше збільшення глибини призвело до перетренування (табл. 4.10).

Таблиця 4.9. Результати порівняльних експериментів моделей БШП із переднавчанням із використанням ГМД та БШП без перднавчання на частково розміченому наборі даних

| Модель | Кількість прихованих шарів | Точність класифікації |
|---|---|---|
| БШП із переднавчанням з ГМД | 1 | 0.8390 |
| | 2 | 0.8521 |
| | 3 | 0.8526 |
| | 4 | 0.8468 |
| БШП без переднавчання | 1 | 0.8380 |
| | 2 | 0.8089 |
| | 3 | 0.7633 |
| | 4 | 0.6620 |

Таблиця 4.10. Результати порівняльних експериментів моделей БШП із переднавчанням із використанням ГМД та БШП без перднавчання на повністю розміченому наборі даних

| Модель | Кількість прихованих шарів | Точність класифікації |
|---|---|---|
| БШП із переднавчанням з ГМД | 1 | 0.9006 |
| | 2 | 0.9042 |
| | 3 | 0.9072 |
| | 4 | 0.9049 |
| БШП без переднавчання | 1 | 0.8872 |
| | 2 | 0.8980 |
| | 3 | 0.8902 |
| | 4 | 0.8731 |

## 4.7. Системні вимоги

Для використання розробленого програмного засобу у формі бібліотеки або програмного додатку необхідно задовольнити системні вимоги, подані у табл. 4.11.

Таблиця 4.11. Системні вимоги для ефективного застосування програмного засобу

| | |
|---|---|
| Оперативна пам’ять, мінімум | 8 ГБ |
| Накопичувач інформації, мінімум | 2 ГБ + обсяг набору даних |
| Графічний адаптер | будь-який CUDA-сумісний |
| Операційна система | така, що підтримує інтерпретатор CPython |

**Висновки по розділу**

Результатом практичної реалізації системи оптимального налаштування ГМД став програмний засіб, що виконує автоматизований пошук оптимальної структури ГМД, оптимальних гіперпараметрів навчання ОМБ, що входять до складу ГМД, утворення з неї БШП із переднавчанням, та остаточного доналаштування для подальшого виробничого використання у складі іншої програмної системи.

# РОЗДІЛ 5. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

## 5.1. Опис ідеї проекту

Опис ідеї стартап-проекту подано у табл. 5.1.

Таблиця 5.1. Опис ідеї стартап-проекту

| Зміст ідеї | Напрямки застосування | Користь для користувача |
|---|---|---|
| Розробка програмного забезпечення, що реалізує оптимальне налаштування глибоких мереж довіри | Робототехнічні системи, що включають компоненти захвату зображення | Автоматичне налаштування нейромережевої моделі для розпізнавання та класифікації образів |
| | Інші програмні системи, що включають нейромережеві моделі, до яких можливе застосування переднавчання | Автоматизація побудови нейромережевих моделей із переднавчанням |

Сильні та слабкі сторони ідеї проекту подано у табл. 5.2.

Таблиця 5.2. Визначення характеристик ідеї проекту

| № п/п | Техніко-економічні характеристики ідеї | Потенційні продукти конкурентів | | | W (слабка сторона) | N (нейтр. сторона) | S (сильна сторона) |
|---|---|---|---|---|---|---|---|
| | | Google Auto ML | H2O.ai | TPOT | | | |
| 1 | Автоматизація налаштування ГМД | - | - | - | | | + |
| 2 | Гнучкість програмної реалізації | + | + | + | | + | |
| 3 | Можливість тонкої конфігурації | - | + | + | | | + |
| 4 | Відомість бренду | + | + | - | - | | |

## 5.2. Технологічний аудит ідеї проекту

Технологічний аудит ідеї проекту наведено у табл. 5.3.

Таблиця 5.3. Технологічна здійсненність ідеї проекту

| № п/п | Складові ідеї проекту | Технології її реалізації | Наявність технологій | Доступність технології |
|---|---|---|---|---|
| 1 | Обробка даних | Python, PyTorch, Numpy | + | Загально-доступні |
| 2 | Організація пошуку оптимальних параметрів моделей у просторі варіантів | Python, PyTorch, Numpy | + | Загально-доступні |
| 3 | Глибоке навчання нейромережевих моделей | PyTorch, CUDA | + | Загально-доступні |
| 4 | Моделювання та аналіз результатів | Python, PyTorch, Numpy, Matplotlib | + | Загально-доступні |

У результаті технологічного аналізу ідеї проекту було вирішено для її реалізації використати наступний технологічний стек:

- Мова програмування та віртуальна машина Python;
- Фреймворк глибокого навчання PyTorch;
- Бібліотека обчислень на графічних адаптерах CUDA;
- Бібліотека тензорних обчислень NumPy;
- Бібліотека візуалізації Matplotlib;
- Фреймворк досліджень Jupyter.

## 5.3. Аналіз ринкових можливостей запуску стартап-проекту

Характеристика потенційного ринку стартап-проекту наведена у табл. 5.4.

Таблиця 5.4. Попередня характеристика потенційного ринку стартап-проекту

| № п/п | Показники стану ринку | Характеристика |
|---|---|---|
| 1 | Кількість конкурентів | 3 |
| 2 | Загальний обсяг продаж | 1 млн. ум. од. |
| 3 | Динаміка ринку (якісна оцінка) | Зростає |
| 4 | Наявність обмежень для входу та їх характер | Немає |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Немає |
| 6 | Середня норма рентабельності по ринку | 10% |

Характеристика потенційних клієнтів стартап-проекту наведена в табл. 5.5.

Таблиця 5.5. Характеристика потенційних клієнтів стартап-проекту

| № п/п | Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності серед різних потенційних цільових груп клієнтів | Вимоги споживачів до продукту |
|---|---|---|---|---|
| 1 | Необхідність сортування сільгосп. продукції | Малий та середній сільськогосподарський бізнес, зайнятий рослинництвом | Масштаби застосування, обсяги оброблюваних даних, технічні засоби для розгортання програмного забезпечення | Ефективність розпізнавання, класифікації та сортування |
| 2 | Необхідність розпізнавання пошкоджень сільгосп. продукції протягом зрощування | | | Ефективність та вчасність розпізнавання пошкоджень с/г рослин на ранніх стадіях |

Можливі загрози для стартап-проекту наведені у табл. 5.6.

Таблиця 5.6. Фактори загроз

| № п/п | Фактор | Зміст загрози | Можлива реакція |
|---|---|---|---|
| 1 | Нестача технічних ресурсів | Малий бізнес може мати дуже обмежені обчислювальні ресурси для розгортання програмного забезпечення | Винесення системи на зовнішні обчислювальні потужності |
| 2 | Відсутність тренувальних наборів даних для моделювання | Замовник може мати технічне забезпечення, проте не мати відповід-ного набору даних | Додавання програмного компонента, що дозволяє напівавтоматично зібрати достатній обсяг даних та частково їх розмітити |
| 3 | Відсутність універсальних інтерфейсів для роботи із системою | Клієнтські інформаційні системи можуть бути не готові до бесшовної інтеграції із продуктом | Розробка інтерфейсів та форматів для інтеграції СОНГМД у клієнтські системи |

Фактори можливостей наведені у табл. 5.7.

Таблиця 5.7. Фактори можливостей

| № п/п | Фактор | Зміст можливості | Можлива реакція |
|---|---|---|---|
| 1 | Хмарні обчислення | Можливість виконання обчислень на віддалених серверах | Пристосування системи для розгортання у хмарних сервісних середовищах |
| 2 | Сторонні сервіси обробки даних | Сторонні сервіси надають послуги розмітки наборів даних на вимогу | Адаптація системи для автоматичної взаємодії із сервісами розмітки наборів даних |

Проведений ступеневий аналіз конкуренції на ринку зображено у табл. 5.8.

Таблиця 5.8. Ступеневий аналіз конкуренції на ринку

| Особливості конкурентного середовища | В чому проявляється дана характеристика | Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною) |
|---|---|---|
| 1. Тип конкуренції – вільна конкуренція | Присутня невелика кількість постачальників. Домінуючих конкурентів немає через галузеву специфіку | Впровадження технологічних інновацій.<br><br>Кооперація з дослідницькими центрами.<br><br>Розширення функціоналу та задоволення потреб клієнтів. |
| 2. За рівнем конкурентної боротьби – глобальний | Продукт не залежить від країни чи локалізації клієнта | |
| 3. За галузевою ознакою – внутрішньогалузева | Продукт спрямований на оптимізації виробничих процесів замовника | |
| 4. Конкуренція за видами товарів: – за необхідністю | Полягає у наданні клієнтові необхідних сервісів | |
| 5. За характером конкурентних переваг – неціновa | Переваги передбачають собою ефективність функціоналу | |
| 6. За інтенсивністю – не марочна | Торгова марка майже немає впливу | |

Проведений аналіз конкуренції в галузі зображено у табл. 5.9.

Таблиця 5.9. Аналіз конкуренції в галузі за М. Портером

| Складові аналізу | Прямі конкуренти в галузі | Потенційні конкуренти |
|---|---|---|
| | Google Cloud AutoML, H2O.ai | TPOT |
| Висновки | Контролюють значну частину ринку, мають узагальнені рішення | Більше розрахований на компанії, що мають спеціалістів із машинного навчання |

Фактори конкурентоспроможності та їх обґрунтування наведені в табл. 5.10.

Таблиця 5.10. Обґрунтування факторів конкурентоспроможності

| № п/п | Фактор | Обґрунтування |
|---|---|---|
| 1 | Інновації | Інноваційні рішення мають забезпечити перевагу нашим клієнтам над конкурентами |
| 2 | Функціонал | Функціонал має покривати вирішення задач клієнтів |
| 3 | Цінова політика | Вартість продукту та його супроводу відіграє велику роль при виборі системи клієнтом |
| 4 | Ресурсоємність | Значні затрати технічних ресурсів можуть створити необхідність залучення додаткових коштів |

Порівняльний аналіз сильних та слабких сторін проекту відображено у табл. 5.11.

Таблиця 5.11. Порівняльний аналіз сильних та слабких сторін RFS

| № п/п | Фактор | Бали 1-20 | Рейтинг продуктів-конкурентів у порівнянні з RFS | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | -3 | -2 | -1 | 0 | +1 | +2 | +3 |
| 1 | Інновації | 17 | | | | + | | | |
| 2 | Функціонал | 12 | + | | | | | | |
| 3 | Цінова політика | 16 | | | + | | | | |
| 4 | Ресурсоємність | 3 | | | | | | + | |

SWOT-аналіз проекту наведено в табл. 5.12.

Таблиця 5.12. SWOT-аналіз стартап-проекту

| | |
|---|---|
| Сильні сторони:<br>розумна цінова політика, функціонал забезпечує рішення більшості задач клієнта | Слабкі сторони:<br>відсутність співпраці з інноваційними центрами |
| Можливості:<br>впровадження інноваційних рішень, оптимізація роботи продукту | Загрози:<br>неточність результатів роботи системи |

Альтернативи ринкового впровадження проекту розглянуто в табл. 5.13.

Таблиця 5.13. Альтернативи ринкового впровадження стартап-проекту

| № п/п | Альтернатива (орієнтовний комплекс заходів) | Ймовірність отримання ресурсів | Терміни реалізації |
|---|---|---|---|
| 1 | Спеціалізовані рішення | Висока | 1-3 місяці |
| 2 | Хмарний сервіс | Висока | 3-6 місяців |
| 3 | Узагальнення рішення, вихід на нові сфери ринку | Середня | 6-12 місяців |

## 5.4. Розробка ринкової стратегії проекту

Опис та вибір цільових груп потенційних клієнтів зображено в табл. 5.14.

Таблиця 5.14. Вибір цільових груп потенційних споживачів

| № п/п | Опис цільової групи потенц. клієнтів | Готовність споживачів сприйняти продукт | Орієнтов-ний попит в межах цільової групи | Інтенсивність конкуренції в сегменті | Простота входу у сегмент |
|---|---|---|---|---|---|
| 1 | Малий бізнес | Висока готовність в розгляді подібних рішень | Високий | Середня | Вхід складний |

| | | | | | |
|---|---|---|---|---|---|
| 2 | Середній бізнес | Середня готовність. В залежності від виду та розташу-вання бізнесу, готовність різниться | Середній | Середня | Вхід середньої складності |
| 3 | Великий бізнес | Низький рівень, оскільки у великий бізнес сприймає комплексні рішення від «іменитих» постачальників «під ключ» | Низький | Висока | Вхід складний |

Обрано цільові групи: 1, 2.

Вибір базової стратегії розвитку подано в табл. 5.15.

Таблиця 5.15. Визначення базової стратегії розвитку

| Обрана альтернатива розвитку проекту | Стратегія охоплення ринку | Ключові конкуренто-спроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
|---|---|---|---|
| Розробка та створення додаткових функціональних модулів | Таргетні пропозиції бізнесу, проведення презентації функціональних рішень на ярмарках та конференціях | Відсутність аналогічних до новостворених функціональних модулів у конкурентів | Розробка та удосконалення існуючих модулів на основі потреб ринку та інформації від клієнтів |

В табл. 5.16 наведено визначення базової стратегії конкурентної поведінки.

Таблиця 5.16. Визначення базової стратегії конкурентної поведінки

| Чи є проект «першопрохід-цем» на ринку? | Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів? | Чи буде компанія копіювати основні характеристики товару конкурента, і які? | Стратегія конкурентної поведінки |
|---|---|---|---|
| Ні | Можливі обидва варіанти | Стандартні функціональні модулі будуть виконувати схожі функції | Унікальна цінова політика, функціональні інновації, сучасні технології |

В табл. 5.17 наведено визначення стратегії позиціонування.

Таблиця 5.17. Визначення стратегії позиціонування

| Вимоги до продукту цільової аудиторії | Базова стратегія розвитку | Ключові конкурентні позиції власного стартап-проекту | Вибір асоціацій, які мають сформувати комплексну позицію власного проекту |
|---|---|---|---|
| Висока якість налаштування нейронних мереж в клієнтській сфері застосування | Розробка та удосконалення існуючих модулів на основі потреб ринку та інформації від клієнтів | Спеціалізовані рішення, хмарні сервіси | Штучний інтелект, адаптивні системи, аналіз даних |

## 5.5. Розроблення маркетингової програми стартап-проекту

У табл. 5.18 представлені ключові переваги концепції потенційного продукту.

Таблиця 5.18. Визначення ключових переваг концепції потенційного продукту.

| № п/п | Потреба | Вигода, яку пропонує продукт | Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити) |
|---|---|---|---|
| 1 | Широкий функціонал | Вирішення задач | Забезпечує вирішення більшої кількості задач бізнесу |
| 2 | Спеціалізовані рішення | Вирішення задач | Забезпечує більш ефективне вирішення задач у звуженій сфері застосування |
| 3 | Технічні ресурси | Забезпечення гнучкості залучення обчислювальних ресурсів | Дозволяє користуватись рішенням за рахунок віддалених технічних потужностей, особливо за умови значної нестачі власних обчислювальних ресурсів |

Визначення меж встановлення ціни показано в табл. 5.19.

Таблиця 5.19. Визначення меж встановлення ціни

| Рівень цін на продукти-замінники | Рівень цін на продукти - аналоги | Рівень доходів цільової групи споживачів | Верхня та нижня межі встановлення ціни на продукт/послугу |
|---|---|---|---|
| - | 200 ум. од. / міс. | Рівень доходів підприємств високий | 100-190 ум.од. / міс. |

Формування системи збуту зображено в табл. 5.20.

Таблиця 5.20. Формування системи збуту

| Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник продукту | Глибина каналу збуту | Оптимальна система збуту |
|---|---|---|---|
| Таргетні пропозиції для компаній | Презентації функціоналу | - | - |

Концепція маркетингових комунікацій відображена в табл. 5.21.

Таблиця 5.21. Концепція маркетингових комунікацій

| № п/п | Специфіка поведінки цільових клієнтів | Канали комунікацій, якими користуються цільові клієнти | Ключові позиції, обрані для позиціонування | Завдання рекламного повідомлення | Концепція рекламного звернення |
|---|---|---|---|---|---|
| 1 | Малий бізнес – оптимальні рішення за невисоку ціну | Соціальні мережі, внутрішньо ринкова комунікація | Прогнозування покупок споживача | Короткий опис переваг продукту, заохочення дізнатись більше | Підвищення ефективності бізнесу клієнтів |
| 2 | Середній бізнес – збалансовані рішення для покращення результатів | Таргетні дзвінки до клієнтів | Прогнозування покупок споживача | Донести інформацію про оптимальність рішення для бізнесу клієнта | Підвищення ефективності бізнесу клієнтів |

**Висновки до розділу**

Відповідно до вищенаведених результатів, можна стверджувати про наявність попиту на запропоновану систему. Варто зауважити, що присутня мала конкуренція, оскільки рішення нове та відносно вузькогалузеве, тож інноваційна складова продукту дозволяє суттєво збільшити конкурентоспроможність проекту.

# ВИСНОВКИ

Розроблено новий алгоритм структурно-параметричного синтезу штучних нейронних мереж глибокого навчання, який базується на використанні нейронних глибоких мереж довіри та багатошарових персептронів, що забезпечує підвищення точності навчання нейронних мереж та розв’язання поставлених перед нею задач.

Розроблено новий метод навчання нейронних глибоких мереж довіри, який базується на оптимальному виборі алгоритму навчання обмеженої машини Больцмана та використанні сучасних алгоритмів оптимізації глибокого навчання.

Розроблено програмний продукт для розв’язання задачі структур структурно-параметричного синтезу нейронних мереж глибокого навчання для застосування у робототехнічних системах.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Hinton, G. E, Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527-1554.

2. Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313:504-507.

3. Goodfellow I., Bengio Y., Courville A. (2016). Deep Learning. MIT Press.

4. Sutskever, I. and Hinton, G. E. (2007) Learning multilevel distributed representations for high-dimensional sequences. AI and Statistics, 2007, Puerto Rico.

5. Chapelle, O., Scholkopf B., Zien, A.. Semi-supervised learning. — Cambridge, Mass.: MIT Press (2006). — ISBN 978-0-262-03358-9.

6. Zhu X. (2017) Semi-supervised Learning. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA

7. Lu, Z., Pu, H., Wang, F., Hu, Z., Wang, L. (2017). The Expressive Power of Neural Networks: A View from the Width. Neural Information Processing Systems, 6231-6239.

8. Liu, C., Zhang, Z., Wang, D.. (2014). Pruning deep neural networks by optimal brain damage. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. 1092-1095.

9. Hinton, G.. (2010). A Practical Guide to Training Restricted Boltzmann Machines. 2010 UTML TR 2010–003, Department of Computer Science, University of Toronto.

10. Carreira-Perpinan, M., Hinton, G.. (2005). On contrastive divergence learning. Artificial Intelligence and Statistics.

11. Hinton, G.. (2002). Training Products of Experts by Minimizing Contrastive Divergence (PDF). Neural Computation. 14 (8): 1771–1800.

12. Tieleman, T.. (2008). Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. Proceedings of the 25th International Conference on Machine Learning. 1064-1071. 10.1145/1390156.1390290.

13. Tieleman, T., Hinton, G.. (2009). Using fast weights to improve persistent contrastive divergence. Proceedings of the 26th International Conference On Machine Learning, ICML 2009. 130. 10.1145/1553374.1553506.

14. Cho, K., Raiko, T., Ilin, A.. (2010). Parallel Tempering is Efficient for Learning Restricted Boltzmann Machines. Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010). 1 - 8. 10.1109/IJCNN.2010. 5596837.

15. Swendsen RH and Wang JS (1986) Replica Monte Carlo simulation of spin glasses Physical Review Letters 57 : 2607–2609

16. Desjardins, Guillaume & Courville, Aaron & Bengio, Y. & Vincent, Pascal & Delalleau, Olivier. (2010). Parallel Tempering for Training of Restricted Boltzmann Machines. 9.

17. Bottou, L.. (1998). "Online Algorithms and Stochastic Approximations". Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6

18. Bottou, L., Bousquet, O.. (2008). The Tradeoffs of Large Scale Learning. Advances in Neural Information Processing Systems. 20. pp. 161–168.

19. Ning Q.. (1998) On the momentum term in gradient descent learning algorithms. Neural networks : the official journal of the International Neural Network Society, 12(1):145–151, 1999.

20. Ruder, S.. (2016). An overview of gradient descent optimization algorithms. https://ruder.io/optimizing-gradient-descent/

21. Нестеров Ю.Е. Метод минимизации выпуклых функций со скоростью сходимости $O(1/k^2)$ // Докл. АН СССР. — 1983. — Т. 269, вып. 3. — С. 543-547.

22. Bengio, Y., Boulanger-Lewandowski, N., Pascanu, R.. (2012). Advances in Optimizing Recurrent Networks. Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on. 10.1109/ICASSP.2013.6639349.

23. Sutskever, I., Martens, J., Dahl, G., Hinton, G.. (2013). On the importance of initialization and momentum in deep learning. 30th International Conference on Machine Learning, ICML 2013. 1139-1147.

24. Duchi, John & Hazan, Elad & Singer, Yoram. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research. 12. 2121-2159.

25. Hinton, G., Tieleman, T.. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: neural networks for machine learning, 4(2):26–31, 2012.

26. Kingma, D., Ba, J.. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations. arXiv:1412.6980.

27. Reddi S., Kale, S., Kumar, S.. (2018). On the Convergence of Adam and Beyond. 2018. arXiv:1904.09237

28. Xavier, G., Bengio, Y.. (2010). Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research - Proceedings Track. 9. 249-256.

29. He, K., Zhang, X., Ren, S., & Sun, J.. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. IEEE International Conference on Computer Vision (ICCV 2015). 1502. 10.1109/ICCV.2015.123.

30. LeCun, Y., Cortes, C., Burges C.J.C.. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/

31. Krizhevsky, A., Nair, V., Hinton G.. The CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html

32. Muresan, H., Oltean, M.. (2018). Fruits-360: A dataset of images containing fruits and vegetables. Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42.

# ДОДАТКИ

# Додаток А

## Блок-схема алгоритму структурно-параметричної побудови глибоких мереж довіри

# Додаток Б

## Структурна схема робототехнічної системи із компонентом оптимального налаштування глибоких мереж довіри

# Додаток В

## Структурна схема системи оптимального налаштування глибоких мереж довіри

# Додаток Г

## Результати порівняльних експериментів по застосуванню алгоритмів CD, PCD та PT для навчання обмеженої машини Больцмана на наборі даних MNIST

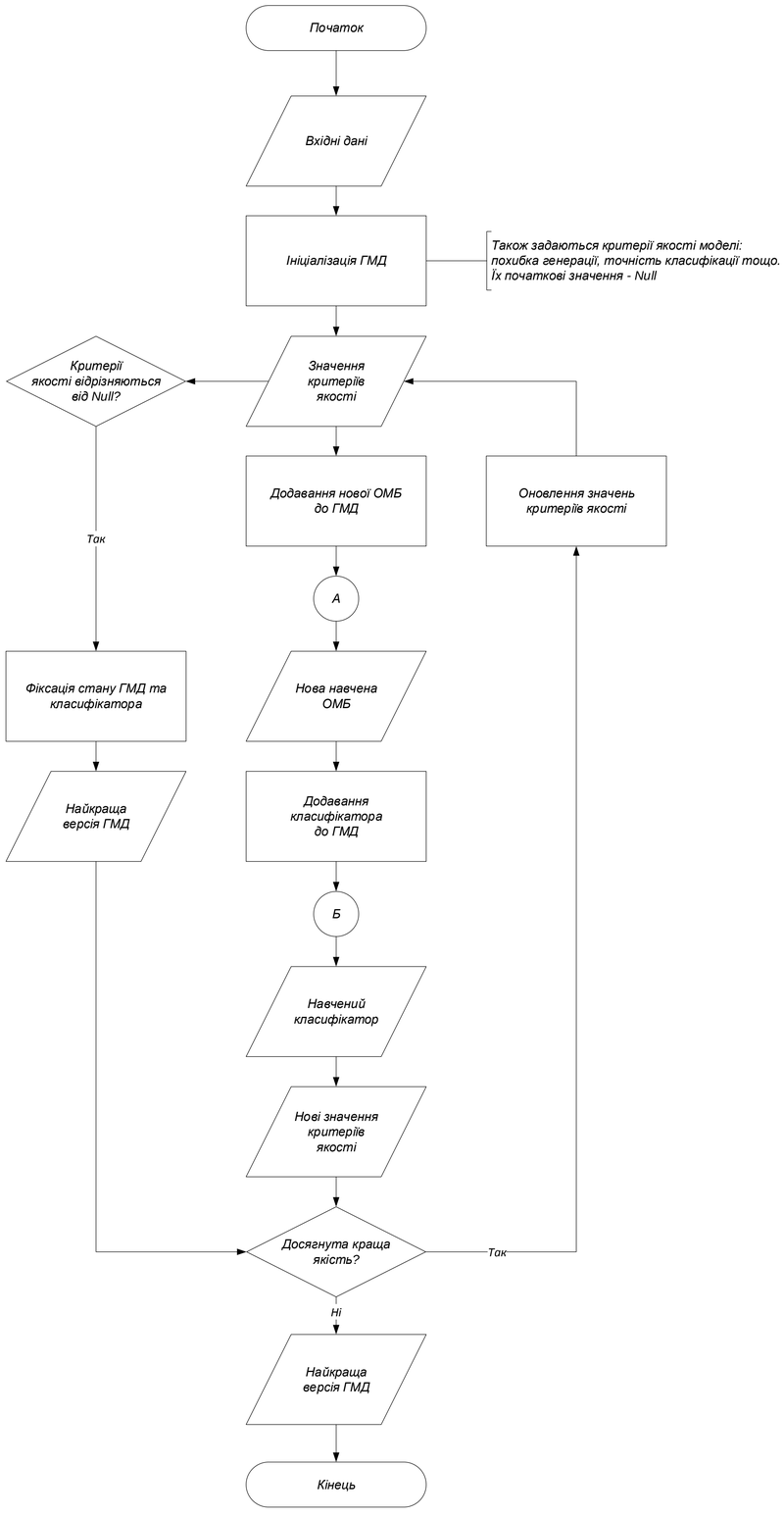
# Додаток Д

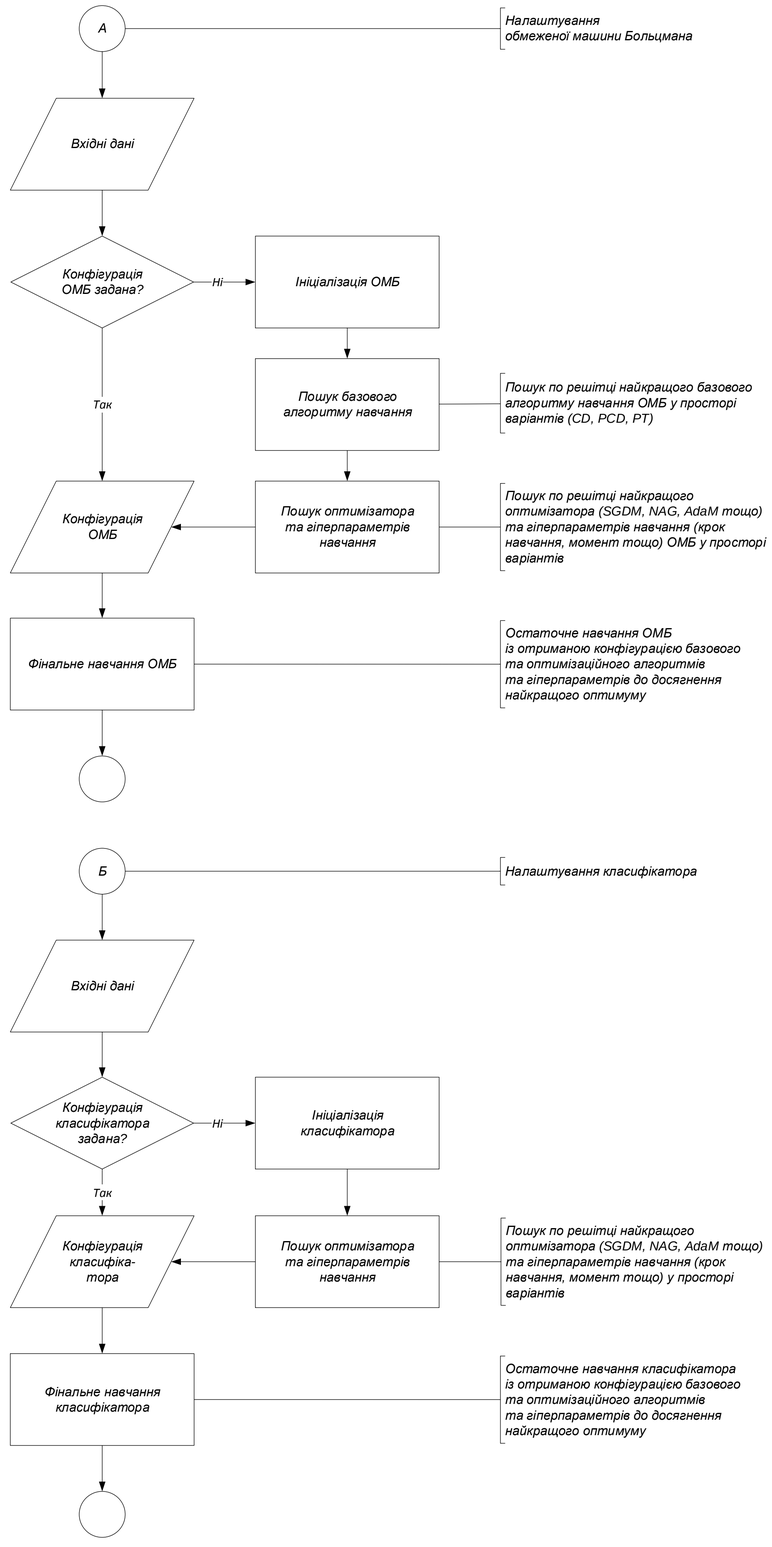
## Результати порівняльних експериментів по застосуванню оптимізаційних алгоритмів для навчання обмеженої машини Больцмана на наборі даних CIFAR-10

# Додаток Е

## Результат перевірки роботи на співпадіння

# Блок-схема алгоритму структурно-параметричної побудови глибоких мереж довіри

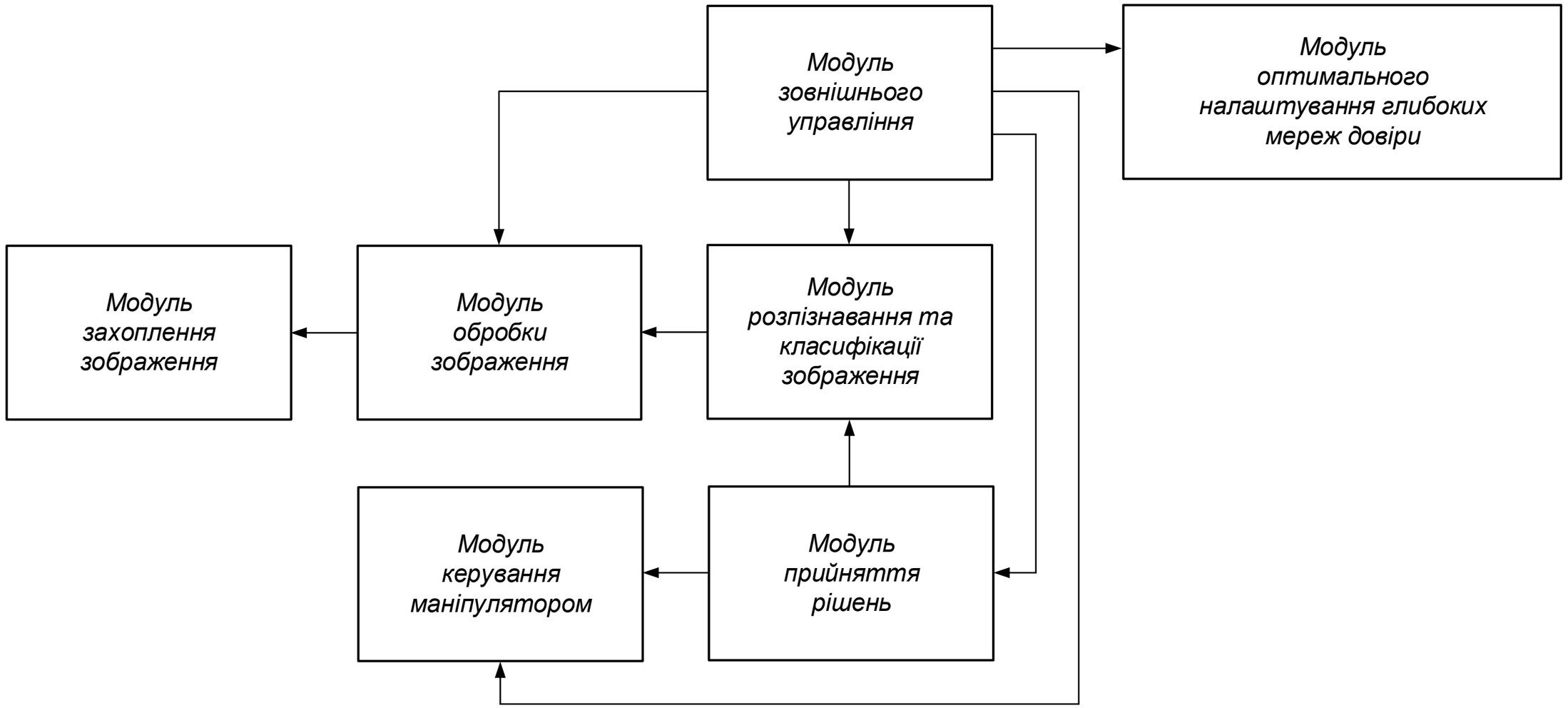



*Демонстраційний плакат № 1*
*до дипломної роботи на тему*
*«Система оптимального налаштування штучних нейронних мереж глибокої довіри»*

*Розробив: Марусик О.М.*
*Прийняла: Чумаченко О.І.*

# *Структурна схема робототехнічної системи із модулем оптимального налаштування глибоких мереж довіри*

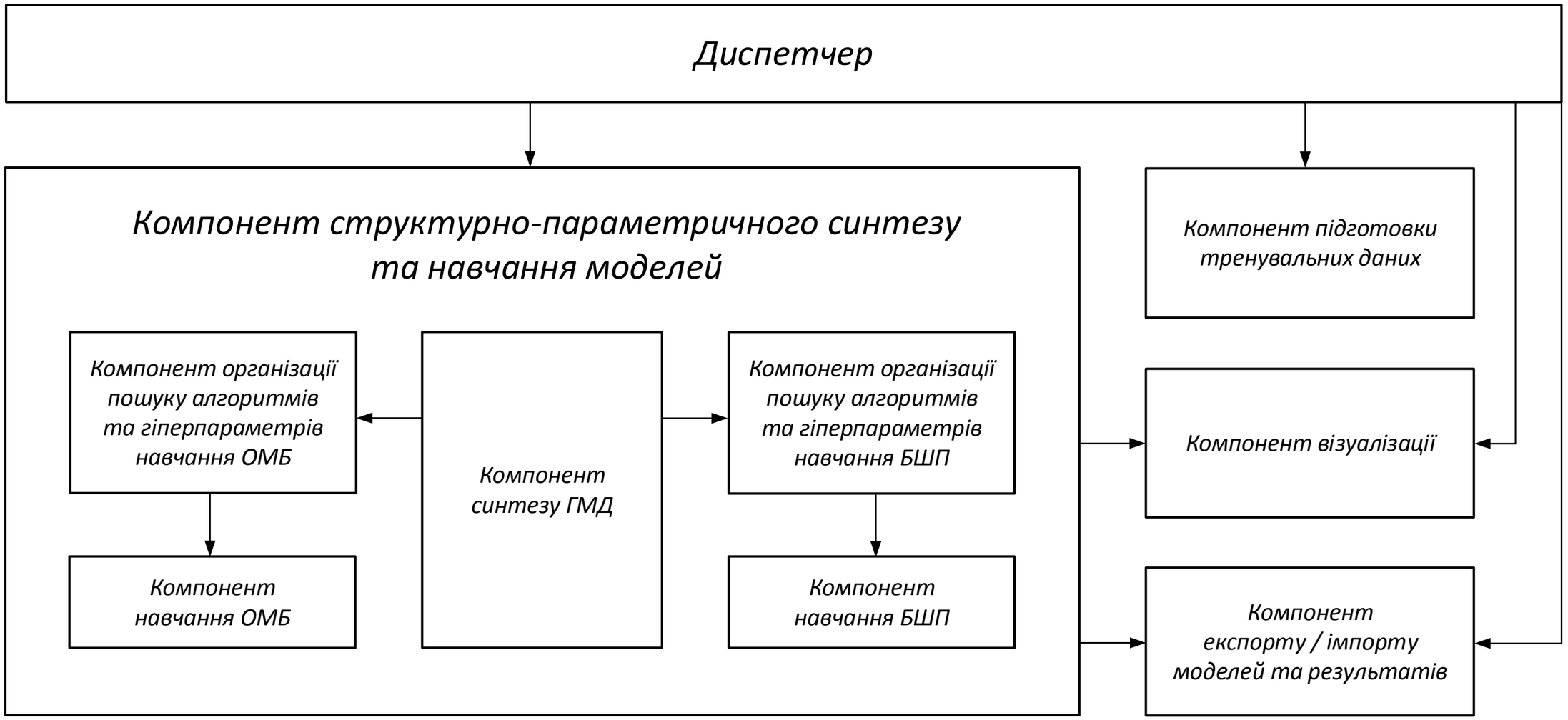



А → Б *Стрілки показують залежність компонентів: «А залежить від Б»*

*Демонстраційний плакат № 2*
*до дипломної роботи на тему*
*«Система оптимального налаштування штучних нейронних мереж глибокої довіри»*

*Розробив: Марусик О.М.*
*Прийняла: Чумаченко О.І.*

# *Структурна схема системи оптимального налаштування глибоких мереж довіри*



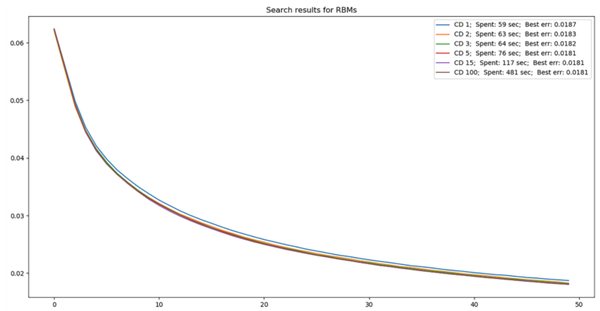

А → Б *Стрілки показують залежність компонентів: «А залежить від Б»*

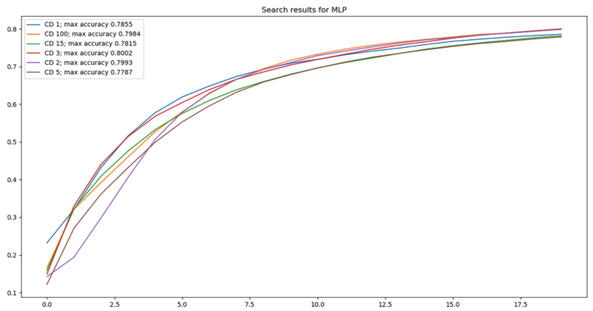
*Демонстраційний плакат № 3*
*до дипломної роботи на тему*
*«Система оптимального налаштування штучних нейронних мереж глибокої довіри»*

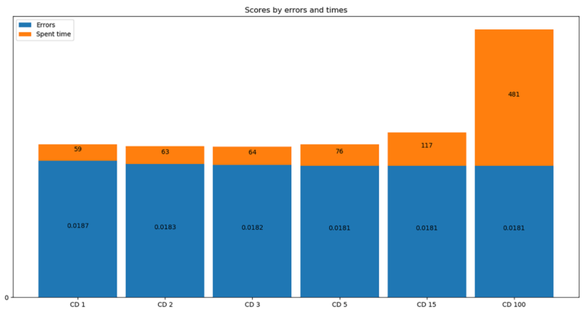
*Розробив: Марусик О.М.*
*Прийняла: Чумаченко О.І.*

# *Результати порівняльних експериментів по застосуванню алгоритмів CD, PCD та PT для навчання обмеженої машини Больцмана на наборі даних MNIST*

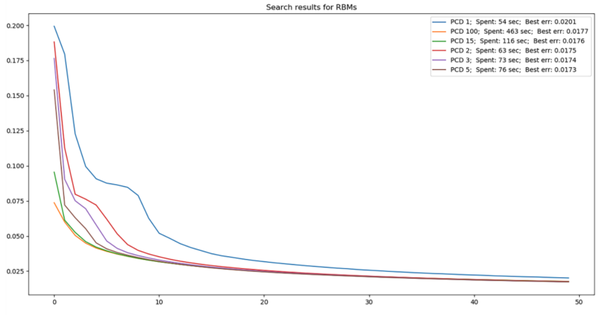
*CD*

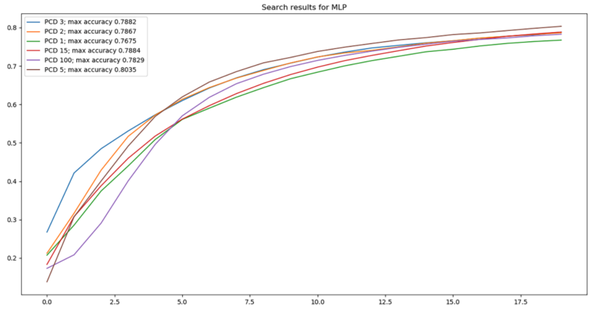


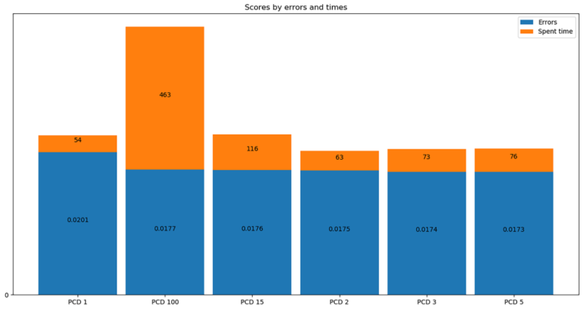



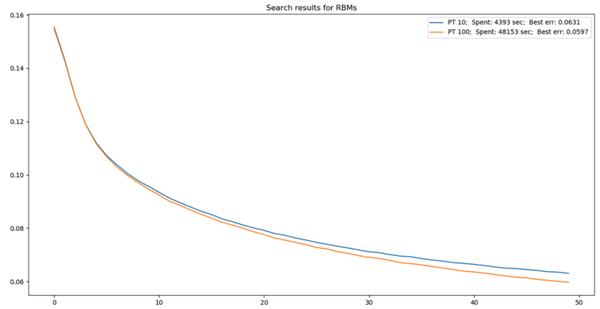

*PCD*

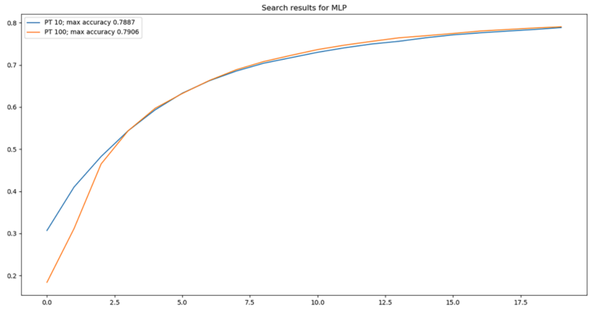


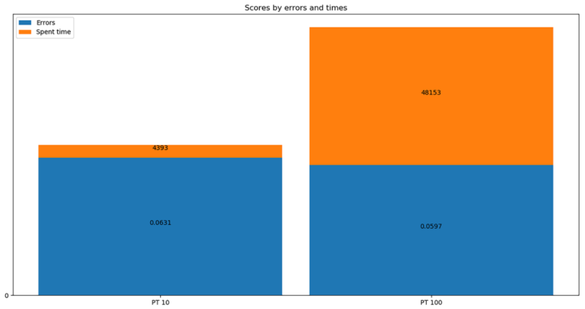



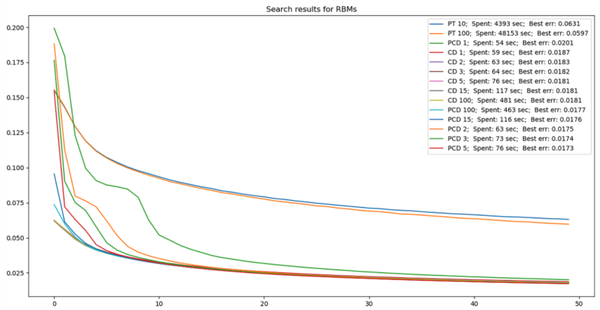

*PT*

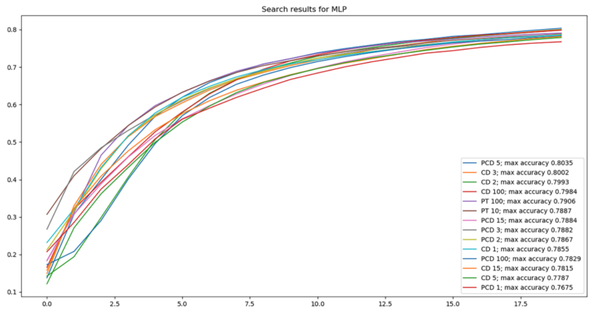


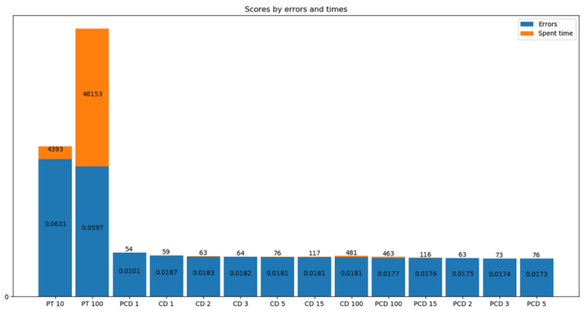




*Разом*




*Похибка генерації* *Точність класифікації* *Оцінка доцільності застосування*

*Демонстраційний плакат № 4*
*до дипломної роботи на тему*
*«Система оптимального налаштування штучних нейронних мереж глибокої довіри»*

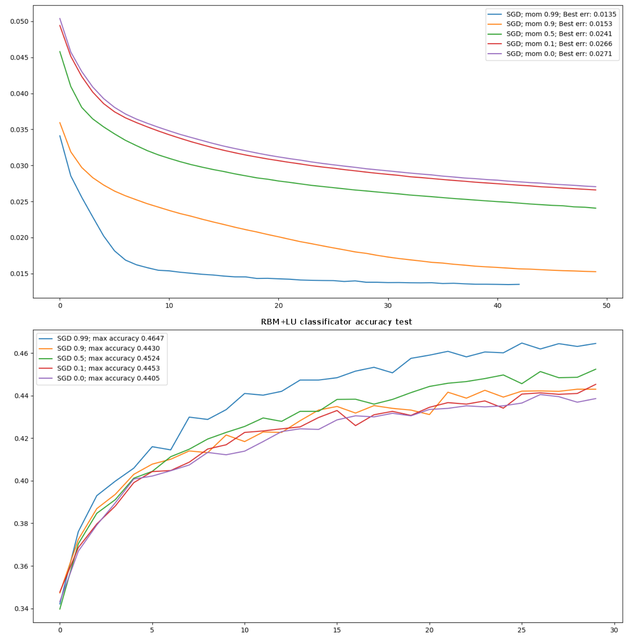
*Розробив: Марусик О.М.*
*Прийняла: Чумаченко О.І.*

# *Результати порівняльних експериментів по застосуванню оптимізаційних алгоритмів для навчання обмеженої машини Больцмана на наборі даних CIFAR-10*

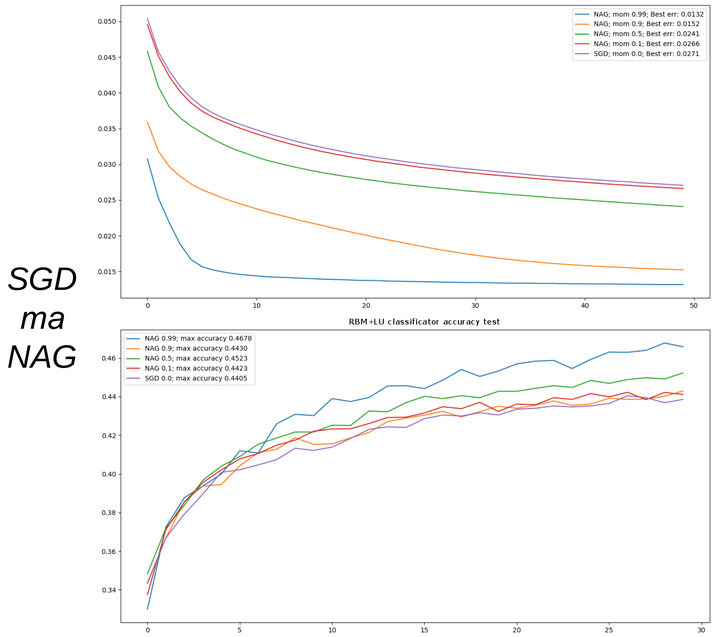
*Похибка генерації*

*Точність класифікації*

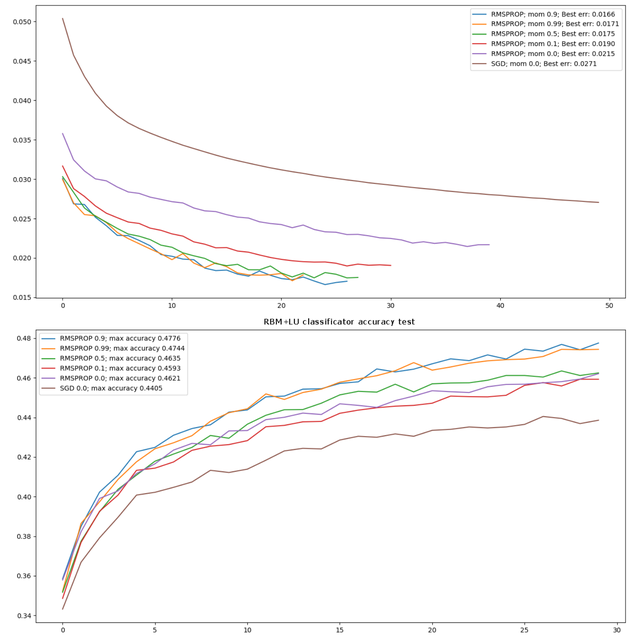
*SGD та SGDM*



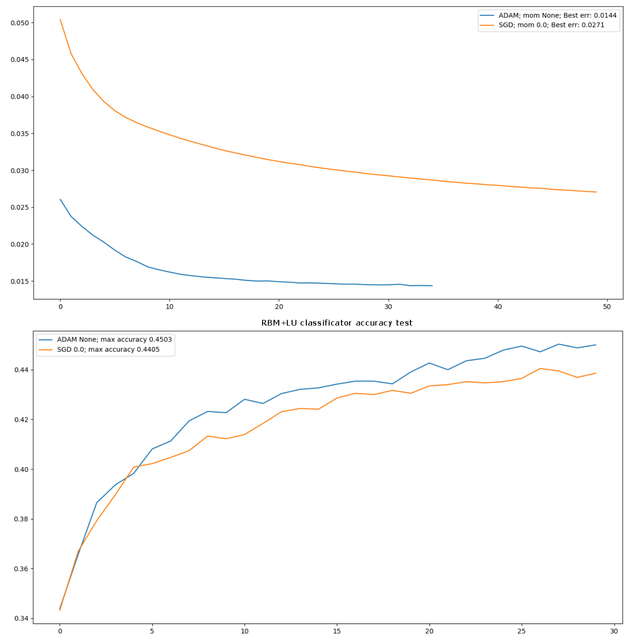

*SGD та NAG*



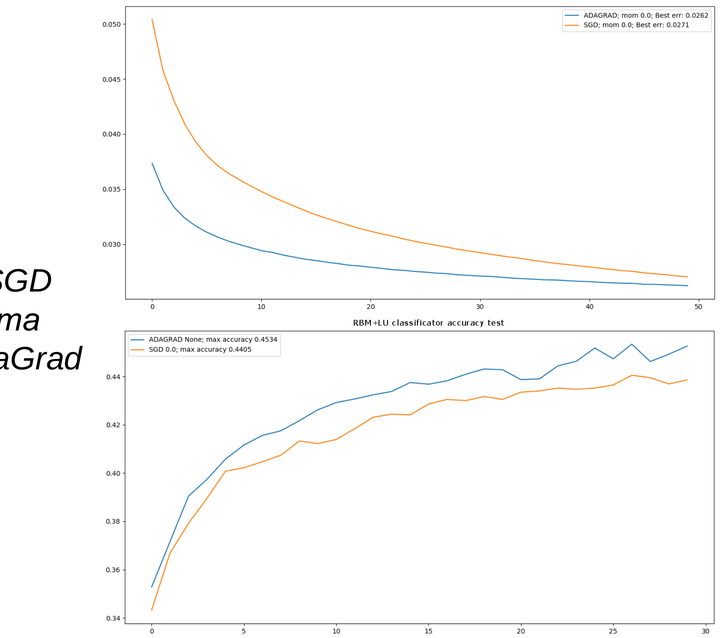

*SGD та RMSProp*



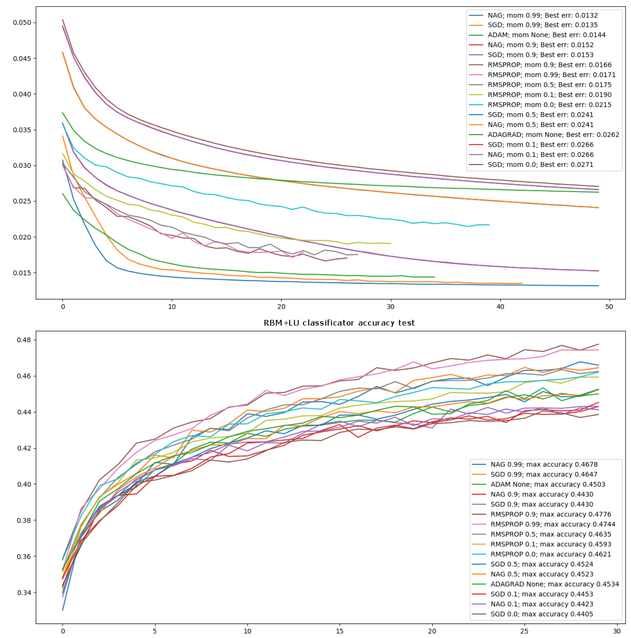

*SGD та AdaM*




*SGD та AdaGrad*




*Усі разом*




*Демонстраційний плакат № 5*
*до дипломної роботи на тему*
*«Система оптимального налаштування штучних нейронних мереж глибокої довіри»*

*Розробив: Марусик О.М.*
*Прийняла: Чумаченко О.І.*